

USER GUIDE

Essential Studio for JavaScript

Version - v24.1.41 | Release Date - December 18, 2023

Browser support	69
Required polyfills	69
Using CDN	69
Node.js	69
Getting started	70
Installation and upgrade	70
Installation	70
Install by using npm CLI.....	70
Install by using package.json.....	70
Download JavaScript – EJ2 Installer	70
Download the Trial Version	71
Installation using Web Installer	73
Overview	74
Installation	74
Uninstallation.....	80
Installation using Offline Installer	85
Installing with UI	85
Installing in silent mode	93
Installing Syncfusion JavaScript – EJ2 Mac Installer.....	94
Steps to resolve the warning message in Catalina OS or later	94
Step-by-Step Installation.....	95
License key registration in samples	99
Linux Installer	99
Download Syncfusion JavaScript Linux Installer	99
Installing Syncfusion JavaScript Linux installer	103
Common Installation Errors	104
Unlocking the license installer using the trial key.....	104
License has expired	104
Unable to find a valid license or trial	105
Unable to install because of another installation	106
Unable to install due to controlled folder access	107
Upgrading Syncfusion JavaScript (Essential JS2).....	109
Upgrading to the latest version	109
Upgrade from trial version to license version.....	109
Licensing.....	109

Appearance	109
Theme in ##Platform_Name## controls.....	109
Reference themes in the application	110
NPM packages.....	110
CDN reference.....	111
Change theme dynamically	113
Common variables	115
Size modes Size Mode for Syncfusion ##Platform_Name## Controls.....	131
Size mode for application	131
Size mode for a control	131
Change size mode for application at runtime.....	132
Change size mode for a control at runtime	133
See also	134
Icons Library.....	134
Referring icons in JavaScript application	134
Steps to use icons library	136
Available icons	138
Overview	139
Customizing theme color from theme studio.....	139
Import previously changed settings into the theme studio	148
Material 3 Theme.....	150
Syncfusion Material 3 Theme	150
What are CSS Variables?	151
Dark mode support	152
ThemeStudio application	155
Common.....	155
Accessibility in Syncfusion ##Platform_Name## controls.....	155
Accessibility overview	155
Accessibility standards.....	155
Accessibility compliance	156
Ensuring accessibility	157
Accessibility support for specific controls.....	157
table	157
Animation in ##Platform_Name##	159
Animation effects.....	159

Animation duration.....	160
Animation delay	161
Enable or disable animation globally	162
Right-To-Left support in Syncfusion JavaScript Controls	162
Enable RTL for all controls.....	162
Enable RTL for an individual control	163
State Persistence in Syncfusion JavaScript controls	164
State Persistence supported controls and properties	167
Template Engine	170
Compiling	170
Available template syntax.....	171
Custom helper.....	171
Template in Syncfusion ##Platform_Name## controls	172
Types of templates.....	172
String template	173
Script template.....	174
Function template.....	176
Getting started with Localization.....	179
Loading translations.....	179
Changing current locale	181
Internationalization.....	182
Loading culture data	182
Changing global culture and currency code.....	182
Manipulating numbers.....	183
Manipulating dateTime.....	187
Drag and drop in ##Platform_Name##.....	194
Draggable	194
Droppable	196
See also	197
Troubleshoot.....	198
Content Security Policy	198
How To	199
Updating Syncfusion npm packages	199
How to load culture files in Essential JS 2 using Ajax.....	199
How to load the culture file in Essential JS 2	202

How to resolve Content Security Policy (CSP) errors.....	204
3D Chart	204
Es5 getting started in EJ2 JavaScript 3D Chart control	204
Dependencies.....	204
Setup for local environment	204
Adding Syncfusion resources	204
Adding 3D Chart control	206
Populate 3D Chart with data.....	207
Add 3D Chart title	210
Enable legend.....	211
Add data label	212
Enable tooltip.....	213
Working with data in EJ2 JavaScript 3D Chart control.....	215
Local data	215
Remote data.....	216
Binding data using ODataAdaptor	217
Empty points	218
Dimensions in EJ2 JavaScript 3D Chart control.....	221
Size for container	221
Size for chart	222
Category axis in EJ2 JavaScript 3D Chart control	224
Labels placement	225
Range	226
Indexed category axis.....	228
Numeric axis in EJ2 JavaScript 3D Chart control.....	229
Range	230
Range padding	231
Label format.....	237
Grouping separator.....	239
Custom label format	240
DateTime axis in EJ2 JavaScript 3D Chart control.....	241
DateTime axis.....	241
DateTime category axis.....	242
Label format.....	249
Custom label format	251

Logarithmic axis in EJ2 JavaScript 3D Chart control.....	252
Range	253
Logarithmic base	255
Logarithmic interval	256
Axis labels in EJ2 JavaScript 3D Chart control	257
Smart axis labels.....	257
Edge label placement.....	261
Maximum labels	262
Axis customization in EJ2 JavaScript 3D Chart control	264
Title	264
Title rotation	265
Tick lines customization	266
Grid lines customization	267
Multiple axis.....	269
Inversed axis.....	270
Opposed position	271
Multiple panes in EJ2 JavaScript 3D Chart control	272
Rows.....	272
Columns	276
Chart Types	279
Column Chart in EJ2 JavaScript 3D Chart control	279
Stacked column chart in EJ2 JavaScript 3D Chart control	285
100% Stacked column chart in EJ2 JavaScript 3D Chart control	290
Bar Chart in EJ2 JavaScript 3D Chart control.....	294
Stacked bar chart in EJ2 JavaScript 3D Chart control	300
100% Stacked bar chart in EJ2 JavaScript 3D Chart control.....	307
Data labels in EJ2 JavaScript 3D Chart control.....	311
Position	312
Template	314
Text mapping	315
Format.....	316
Margin.....	317
Customization	319
Customizing specific label	320
Legend in EJ2 JavaScript 3D Chart control	321

Position and alignment	321
Legend customization	327
Series selection through legend.....	334
Collapsing legend item	336
Legend title	337
Arrow page navigation.....	339
Legend item padding	340
Tooltip in EJ2 JavaScript 3D Chart control	342
Default tooltip.....	342
Fixed tooltip	343
Format the tooltip.....	344
Tooltip template	346
Customize the appearance of tooltip	347
Selection in EJ2 JavaScript 3D Chart control.....	348
Point.....	349
Series.....	350
Cluster	351
Selection type.....	352
Selection during initial loading.....	354
Selection through legend.....	355
Print and Export in EJ2 JavaScript 3D Chart control	356
Print.....	356
Export.....	358
Appearance in EJ2 JavaScript 3D Chart control	359
Custom color palette.....	359
Data point customization.....	360
Point level customization.....	361
Chart area customization.....	363
Animation.....	365
Chart rotation.....	366
Title	367
Accessibility in EJ2 JavaScript 3D Chart control	373
WAI-ARIA.....	373
Keyboard navigation	374
Accordion	374

Expand mode in ##Platform_Name## Accordion control	374
Single	374
Multiple	375
See Also	377
Accessibility in ##Platform_Name## Accordion control	377
ARIA attributes	378
Keyboard interaction	378
Ensuring accessibility	379
See also	379
Style in ##Platform_Name## Accordion control	379
Customizing Accordion	379
Customizing the list items	379
Customizing Accordion's header	379
Customizing Accordion's expand and collapse icons	380
Customizing the hover state of Accordion control	380
Customizing selected item of Accordion control	380
How To	380
Set the nested accordion in ##Platform_Name## Accordion control	380
Load content through post in ##Platform_Name## Accordion control	382
Set custom animation in ##Platform_Name## Accordion control	384
To keep single pane open always in ##Platform_Name## Accordion control	386
Create wizard using accordion in ##Platform_Name## Accordion control	388
Load accordion with data source in ##Platform_Name## Accordion control	393
Load accordion items dynamically in ##Platform_Name## Accordion control	394
Add icon to accordion header in ##Platform_Name## Accordion control	397
Add font awesome in ##Platform_Name## Accordion control	404
Customize expand collapse actions in ##Platform_Name## Accordion control	406
Integrate treeview inside the accordion in ##Platform_Name## Accordion control	408
AccumulationChart	411
Pie dough nut in ##Platform_Name## Accumulation chart control	411
Pie Chart	411
Radius Customization	412
Pie Center	413
Various Radius Pie Chart	414
Doughnut Chart	415

Start and End angles	416
Color & Text Mapping	417
Customization	418
Hide pie or doughnut border	419
Multi-level pie chart.....	420
See Also	424
Pyramid in ##Platform_Name## Accumulation chart control.....	424
Mode	425
Size	426
Gap Between the Segments.....	427
Explode.....	428
Customization	429
See Also	430
Funnel in ##Platform_Name## Accumulation chart control.....	430
Size	431
Neck Size	432
Gap Between the Segments.....	433
Explode.....	434
Smart Data Label.....	436
Customization	437
See Also	438
Data label in ##Platform_Name## Accumulation chart control.....	438
Positioning.....	440
Smart labels.....	441
Data Label Template	442
Connector Line	443
Text Mapping	444
Format.....	445
Customization	447
Text wrap	448
Show percentages in data labels of pie chart	449
Grouping in ##Platform_Name## Accumulation chart control.....	451
Group Mode.....	453
Customization	454
Empty points in ##Platform_Name## Accumulation chart control	455

Customization	456
Annotation in ##Platform_Name## Accumulation chart control.....	457
Region	458
Co-ordinate Units.....	459
Alignment.....	461
Tooltip in ##Platform_Name## Accumulation chart control	462
Header.....	463
Format.....	464
Tooltip format	465
Fixed tooltip	466
Customization	467
To customize individual tooltip.....	468
Legend in ##Platform_Name## Accumulation chart control	469
Position and Alignment.....	470
Legend Reverse	471
Legend Shape	472
Legend Size.....	474
Legend Item Size	475
Paging for Legend.....	476
Legend Text Wrap	477
Legend Title.....	478
Arrow Page Navigation	479
Legend Item Padding	480
Center label in ##Platform_Name## Accumulation chart control	481
Hover text	482
Customization	483
Title and sub title in ##Platform_Name## Accumulation chart control.....	485
Title Customization	486
SubTitle	487
SubTitle Customization	488
Chart print in ##Platform_Name## Accumulation chart control	489
Print.....	489
Export.....	490
Accessibility in ##Platform_Name## Accumulation chart control	491
WAI-ARIA attributes.....	492

Keyboard interaction	493
Ensuring accessibility	493
See also	493
Ej1 api migration in ##Platform_Name## Accumulation chart control.....	493
Accumulation Chart	493
Annotation	496
Series.....	497
DataLabel	502
Legend.....	504
Methods.....	506
Events.....	507
AppBar.....	510
Size and color in ##Platform_Name## AppBar control	510
Size	510
Color.....	514
Accessibility in ##Platform_Name## AppBar control.....	519
Keyboard interaction	520
Ensuring accessibility	520
See also	520
Position in ##Platform_Name## AppBar control.....	521
Top AppBar	521
Bottom AppBar	522
Sticky AppBar	524
Design in ##Platform_Name## AppBar control.....	526
Spacer.....	526
Separator.....	527
Media Query	529
Designing AppBar with Menu	530
Designing AppBar with Buttons	532
Designing AppBar with SideBar.....	534
Style and appearance in ##Platform_Name## AppBar control.....	536
CssClass	537
HtmlAttributes	538
AutoComplete	539
Data binding in ##Platform_Name## Auto complete control.....	539

Bind to local data	539
Bind to remote data	543
See Also	544
Templates in ##Platform_Name## Auto complete control	544
Item template	544
Group template	546
Header template	547
Footer template	548
No records template	549
Action failure template	550
See Also	552
Grouping in ##Platform_Name## Auto complete control	552
Customization	553
See Also	553
Filtering in ##Platform_Name## Auto complete control	553
Change the filter type	553
Filter item count	555
Limit the minimum filter character	556
Case sensitive filtering	557
Diacritics Filtering	558
See Also	560
Virtualization in AutoComplete Component	560
Binding local data	560
Binding Remote data	561
Grouping with Virtualization	563
Localization in ##Platform_Name## Auto complete control	564
Loading translations	564
See Also	566
Style in ##Platform_Name## Auto complete control	566
Customizing the appearance of wrapper element	566
Customizing the dropdown icon's color	566
Customizing the focus color	567
Customizing the outline theme's focus color	567
Customizing the float label element's focusing color	567
Customizing the color of the placeholder text	568

Customizing the text selection color.....	568
Customizing the background color of focus, hover, and active item's	568
Customizing the appearance of pop-up element	568
Adding mandatory asterisk to placeholder and float label.....	569
Accessibility in <code>##Platform_Name##</code> Auto complete control.....	570
WAI-ARIA attributes.....	571
Keyboard interaction	571
Ensuring accessibility	573
See also	573
How To	573
Autofill in <code>##Platform_Name##</code> Auto complete control.....	573
Icon support in <code>##Platform_Name##</code> Auto complete control	575
Custom search in <code>##Platform_Name##</code> Auto complete control.....	576
Filter in <code>##Platform_Name##</code> Auto complete control	577
Achieve virtual scrolling in <code>##Platform_Name##</code> Auto complete control	579
Avatar.....	581
Types in <code>##Platform_Name##</code> Avatar control	581
Avatar size	581
Avatar types	582
How To	584
Avatar customization in <code>##Platform_Name##</code> Avatar control.....	584
Integrate avatar into listview in <code>##Platform_Name##</code> Avatar control	588
Integrate avatar into badge in <code>##Platform_Name##</code> Avatar control.....	589
Badge	592
Types in <code>##Platform_Name##</code> Badge control	592
Badge styles	592
Badge types.....	595
How To	603
Badge customization in <code>##Platform_Name##</code> Badge control.....	603
Integrate badge into listview in <code>##Platform_Name##</code> Badge control	607
Dynamic badge content in <code>##Platform_Name##</code> Badge control	608
Barcode	610
Getting started in <code>##Platform_Name##</code> Barcode control.....	610
Dependencies.....	610
Setup for local development.....	611

Configuring system JS	611
Adding CSS reference.....	612
Adding Barcode Generator control.....	612
Adding QR Generator control	614
Adding Datamatrix Generator control	615
BarcodeGenerator in ##Platform_Name## Barcode control	616
Code39	616
Code39 Extended	617
Code 11	618
Codabar	619
Code 32	620
Code 93	621
Code 93 Extended	622
Code 128	622
Customizing the Barcode color	623
Customizing the Barcode dimension	624
Customizing the text	625
Qrcodegenerator in ##Platform_Name## Barcode control	626
QR Code	626
Customizing the Barcode color	628
Customizing the Barcode dimension	629
Customizing the text	630
Datamatrixgenerator in ##Platform_Name## Barcode control	631
Data Matrix	631
Customizing the Barcode color	632
Customizing the Barcode dimension	633
Customizing the text	634
Export in ##Platform_Name## Barcode control.....	635
Export.....	635
Breadcrumb	636
Data binding in ##Platform_Name## Breadcrumb control	636
Items based on current Url	636
Absolute Url	637
Customize text when generated items using Url.....	638
Icons in ##Platform_Name## Breadcrumb control	639

Icon in Breadcrumb item	639
Icon Position.....	643
Icon Only	644
Show icon only for first item.....	645
Navigation in ##Platform_Name## Breadcrumb control	647
URL	647
Enable navigation for last Breadcrumb item	649
Open URL in new page or tab	651
Templates in ##Platform_Name## Breadcrumb control.....	652
Item Template.....	652
Separator Template	654
Customize Specific Item Template.....	655
Overflow in ##Platform_Name## Breadcrumb control.....	656
Collapsed.....	658
Menu.....	660
Wrap.....	661
Scroll.....	663
Hidden.....	664
None.....	666
Accessibility in ##Platform_Name## Breadcrumb control	666
WAI-ARIA attributes.....	667
Keyboard interaction	667
Ensuring accessibility	667
See also	667
Bullet Chart	667
Bullet chart dimensions in ##Platform_Name## Bullet chart control.....	667
Size for Container.....	667
Size for Bullet Chart.....	668
Axis customization in ##Platform_Name## Bullet chart control.....	670
MajorTickLines and MinorTickLines Customization.....	670
Tick Placement	671
Label Format	672
GroupingSeparator	673
Custom Label Format.....	674
Label Placement.....	675

Opposed Position	676
Category Label	677
Category Label Customization	678
Data binding in ##Platform_Name## Bullet chart control	679
Ranges in ##Platform_Name## Bullet chart control	680
Color Customization.....	681
Value bar in ##Platform_Name## Bullet chart control	682
Types of actual bar	683
Actual bar customization	684
Comparative bar in ##Platform_Name## Bullet chart control.....	686
Types of target bar	687
Target bar customization	688
Title in ##Platform_Name## Bullet chart control.....	689
Title	689
Subtitle	690
Title and SubTitle Position	691
Title Customization	695
SubTitle Customization	696
Customization in ##Platform_Name## Bullet chart control.....	697
Orientation.....	697
Right-to-left (RTL).....	698
Animation.....	699
Theme	700
Data label in ##Platform_Name## Bullet chart control	701
Data Label Customization	702
Tool tip in ##Platform_Name## Bullet chart control	704
Default tooltip.....	704
Tooltip template	705
Customization of the appearance of tooltip.....	707
Accessibility in ##Platform_Name## Bullet chart control	708
WAI-ARIA attributes.....	709
Keyboard interaction	709
Ensuring accessibility	710
See also	710
ButtonGroup	710

Getting started in ##Platform_Name## Button group control	710
Dependencies.....	710
Set up development environment	710
Add Syncfusion JavaScript packages	710
Import the Syncfusion CSS styles	711
Add ButtonGroup to the project.....	711
Run the application	712
Orientation.....	713
See Also	714
Types and styles in ##Platform_Name## Button group control.....	714
ButtonGroup types	714
ButtonGroup styles	715
See Also	717
Selection in ##Platform_Name## Button group control	717
Selection.....	717
Nesting	720
See Also	723
Accessibility in ##Platform_Name## Button group control	723
Keyboard interaction	724
Ensuring accessibility	724
See also	725
How To	725
Create buttongroup with icons in ##Platform_Name## Button group control	725
Create buttongroup with rounded corner in ##Platform_Name## Button group control	726
Disable in ##Platform_Name## Button group control	727
Enable ripple in ##Platform_Name## Button group control.....	729
Enable rtl in ##Platform_Name## Button group control.....	730
Form submit in ##Platform_Name## Button group control	731
Initialize buttongroup using util function in ##Platform_Name## Button group control.....	733
Show buttongroup selected state on initial render in ##Platform_Name## Button group control	735
Button	736
Types and styles in ##Platform_Name## Button control.....	736
Button styles	736
Button types.....	738
Icons.....	744

Button size	748
See Also	749
Accessibility in ##Platform_Name## Button component.....	749
WAI-ARIA attributes.....	750
Keyboard interaction	750
Ensuring accessibility	750
See also	750
How To	751
Add link to a button in ##Platform_Name## Button control	751
Create a block button in ##Platform_Name## Button control	752
Customize button appearance in ##Platform_Name## Button control.....	753
Customize input and anchor elements in ##Platform_Name## Button control	755
Repeat button in ##Platform_Name## Button control.....	756
Right to left in ##Platform_Name## Button control	759
Set the disabled state in ##Platform_Name## Button control	761
Tooltip for button in ##Platform_Name## Button control	763
Calendar	764
Date range in ##Platform_Name## Calendar control	764
Multi select in ##Platform_Name## Calendar control	766
See Also	767
Globalization in ##Platform_Name## Calendar control.....	767
Right-to-left.....	772
Customization in ##Platform_Name## Calendar control	774
Disable weekends	774
Day cell format.....	775
Highlight weekends.....	777
See Also	778
Calendar views in ##Platform_Name## Calendar control.....	778
View restriction.....	780
Accessibility in ##Platform_Name## Calendar control	781
WAI-ARIA attributes.....	782
Keyboard interaction	782
Ensuring accessibility	784
See also	784
Islamic calendar in ##Platform_Name## Calendar control	784

Style appearance in ##Platform_Name## Calendar control	786
Customizing the background color for the Calendar	786
Customizing the Calendar date elements on hovering.....	786
Customizing the border of date cell grid	786
Customizing the Calendar title.....	787
Customizing the previous and next icon.....	787
Customizing the footer button	787
Customizing the selected date cell grid	787
Customizing the content header in Calendar	788
How To	788
Set clear button in calendar in ##Platform_Name## Calendar control	788
Show dates of other months in ##Platform_Name## Calendar control	789
Select a sequence of dates in calendar in ##Platform_Name## Calendar control	791
Skip a month in calendar in ##Platform_Name## Calendar control	793
Render the calendar with week numbers in ##Platform_Name## Calendar control	794
Change the first day of week in ##Platform_Name## Calendar control	795
Customize the calendar day header in ##Platform_Name## Calendar control	796
Card.....	797
Header content in ##Platform_Name## Card control.....	797
Header.....	797
Content	799
Card image in ##Platform_Name## Card control.....	800
Images	800
Divider	801
See Also	802
Action buttons in ##Platform_Name## Card control	802
Vertical	803
See Also	804
Horizontal in ##Platform_Name## Card control	804
Stacked cards	805
Style in ##Platform_Name## Card control	806
Customizing the card	806
Customizing the Header element	806
Customizing the card content.....	806
Divider used to separate the elements inside the card	807

Including image within card element	807
Including a title or caption for the image	807
To include heading image within the header	807
Customizing the Header main title	808
Customizing the Header subtitle.....	808
Including action buttons or anchor tags	808
To align card elements horizontally	808
To align elements vertically within the horizontal layout.....	809
How To	809
Customize the card image title position in ##Platform_Name## Card control.....	809
Integrate other component inside the card in ##Platform_Name## Card control	810
Carousel	811
Populating items in ##Platform_Name## Carousel control	811
Populating items using carousel item	811
Populating items using data source	813
Selection.....	814
Partial visible slides	817
See Also	820
Navigators and indicators in ##Platform_Name## Carousel control	820
Navigators	820
Indicators	825
Play button.....	835
Animations	839
Intervals between slides	842
Auto play slides	843
Pause on hover.....	844
Looping slides.....	846
Slide changing events.....	847
Disable touch swiping	849
Swipe Modes.....	850
Accessibility in ##Platform_Name## Carousel control.....	852
ARIA attributes.....	853
Keyboard interaction	853
Ensuring accessibility	854
See also	854

Styles and appearance in ##Platform_Name## Carousel control	854
CSS Structure in JavaScript Carousel Control.....	854
Customizing the indicators	855
Customizing the navigators.....	857
Customizing partial slides size	860
Chart.....	861
Working with data in ##Platform_Name## Chart control.....	861
Local Data.....	861
Remote Data	862
Binding Data Using ODataAdaptor.....	863
Lazy loading.....	864
Empty points	866
Chart dimensions in ##Platform_Name## Chart control	869
Size for Container.....	869
Size for Chart.....	870
Category axis in ##Platform_Name## Chart control	872
Labels Placement	874
Range	875
Indexed category axis.....	876
Numeric axis in ##Platform_Name## Chart control.....	878
Range	879
Range Padding.....	880
Label Format	886
GroupingSeparator	888
Custom Label Format.....	888
Date time axis in ##Platform_Name## Chart control.....	890
DateTime Axis	890
DateTimeCategory Axis.....	891
Label Format	898
Custom Label Format.....	899
Logarithmic axis in ##Platform_Name## Chart control.....	901
Range	902
Logarithmic Base.....	903
Logarithmic Interval	905
Axis labels in ##Platform_Name## Chart control.....	906

Smart Axis Labels	906
Axis Labels Positioning	910
Multilevel Labels	911
Edge Label Placement	918
Sorting	919
Labels Customization	921
Customizing Specific Point	922
Line break support	923
Maximum Labels	925
Axis customization in ##Platform_Name## Chart control	926
Axis Crossing	926
Title	928
Title Rotation.....	929
Tick Lines Customization	930
Grid Lines Customization	932
Multiple Axis	933
Inversed Axis	935
Opposed Position	936
Strip line in ##Platform_Name## Chart control	937
Horizontal Strip lines.....	937
Vertical Striplines	939
Customize the strip line	940
Customize the stripline text.....	941
See Also	942
Multiple panes in ##Platform_Name## Chart control	942
Rows.....	942
Columns	946
Chart Types	949
Line Chart in ##Platform_Name## control	949
Step line Chart in ##Platform_Name## control.....	953
Stack line Chart in ##Platform_Name## control	955
Stacked line Chart in ##Platform_Name## control	958
Spline Chart in ##Platform_Name## control.....	962
Area Chart in ##Platform_Name## control	964
Range area Chart in ##Platform_Name## control.....	969

Range step area Chart in ##Platform_Name## control.....	972
Spline range area Chart in ##Platform_Name## control.....	975
Stack area Chart in ##Platform_Name## control	978
Stacked area Chart in ##Platform_Name## control	981
Stacked step area Chart in ##Platform_Name## control	984
Step area Chart in ##Platform_Name## control	988
Spline area Chart in ##Platform_Name## control.....	990
Column Chart in ##Platform_Name## control	993
Range column Chart in ##Platform_Name## control.....	999
Stack column Chart in ##Platform_Name## control	1002
Stacked column Chart in ##Platform_Name## control	1008
Bar Chart in ##Platform_Name## control	1013
Stack bar Chart in ##Platform_Name## control.....	1019
Stacked bar Chart in ##Platform_Name## control.....	1025
Scatter Chart in ##Platform_Name## control	1029
Bubble Chart in ##Platform_Name## control	1033
Polar Chart in ##Platform_Name## control	1037
Radar Chart in ##Platform_Name## control	1050
High low Chart in ##Platform_Name## control.....	1054
High low open close Chart in ##Platform_Name## control	1057
Candle Chart in ##Platform_Name## control.....	1059
Box whisker Chart in ##Platform_Name## control	1062
Waterfall Chart in ##Platform_Name## control.....	1066
Histogram Chart in ##Platform_Name## control	1069
Error bar Chart in ##Platform_Name## control	1071
Vertical Chart in ##Platform_Name## control	1078
Pare to Chart in ##Platform_Name## control	1080
Polar radar in ##Platform_Name## Chart control.....	1083
Polar Chart	1083
Radar Chart	1093
See Also	1097
Chart series in ##Platform_Name## Chart control.....	1097
Multiple Series	1097
Combination Series	1099
Enable Complex Property in Series	1100

Technical indicators in ##Platform_Name## Chart control.....	1102
Accumulation Distribution	1102
Average True Range (ATR)	1106
Bollinger Band	1109
Exponential Moving Average (EMA)	1116
Momentum	1119
Moving Average Convergence Divergence (MACD)	1126
Relative Strength Index (RSI).....	1133
Simple Moving Average (SMA)	1137
Stochastic	1140
Triangular Moving Average (TMA).....	1147
Trend lines in ##Platform_Name## Chart control.....	1157
Linear.....	1158
Exponential	1159
Logarithmic	1160
Polynomial	1162
Power	1163
Moving Average	1164
Forecasting.....	1167
Show or hide a trendline.....	1169
Data markers in ##Platform_Name## Chart control.....	1171
Marker.....	1171
Shape.....	1172
Images	1174
Customization	1175
Customizing specific point	1177
Fill marker with series color	1178
See also	1180
Data labels in ##Platform_Name## Chart control.....	1180
Position	1182
Data Label Template	1183
Text Mapping	1184
Format.....	1186
Margin.....	1187
Customization	1189

Customizing Specific Point	1190
Show percentage based on each series points.....	1192
See Also	1193
Chart annotations in ##Platform_Name## Chart control	1193
Region	1195
Co-ordinate Units.....	1196
Alignment.....	1198
Adding y-axis sub title through on annotation	1199
See Also	1201
Legend in ##Platform_Name## Chart control	1201
Position and Alignment.....	1201
Customization	1206
Series selection on Legend.....	1215
Collapsing Legend Item	1216
Legend Title.....	1218
Arrow Page Navigation	1219
Legend Item Padding	1221
See Also	1222
Tooltip in ##Platform_Name## Chart control	1222
Default tooltip.....	1222
Fixed tooltip	1224
Format the tooltip.....	1226
Tooltip template	1228
Customize the appearance of tooltip	1230
See also	1232
Zooming in ##Platform_Name## Chart control.....	1232
Enable zooming.....	1232
Modes	1234
Toolbar	1236
Enable pan.....	1237
Enable scrollbar.....	1239
Auto interval on zooming.....	1241
Data editing in ##Platform_Name## Chart control	1243
Enable Data Editing.....	1243
Cross hair and track ball in ##Platform_Name## Chart control.....	1245

Tooltip for axis	1247
Customization	1249
Trackball	1251
Synchronized Charts in ##Platform_Name## Chart control	1253
Tooltip synchronization	1253
Crosshair synchronization	1256
Zooming synchronization	1259
Selection synchronization	1262
Selection in ##Platform_Name## Chart control	1265
Point	1265
Series	1267
Cluster	1268
Rectangular selection	1270
Selection type	1272
Selection on load	1273
Selection through on legend	1275
Customization for selection	1276
See Also	1278
Chart print in ##Platform_Name## Chart control	1278
Print	1278
Export	1279
Multiple chart export	1285
Exporting chart using base64 string	1287
Chart appearance in ##Platform_Name## Chart control	1289
Custom color palette	1289
Data point customization	1290
Point level customization	1293
Chart area customization	1294
Animation	1298
Chart title	1301
See also	1309
Render methods in ##Platform_Name## Chart control	1309
Canvas	1309
Accessibility in ##Platform_Name## Chart control	1309
WAI-ARIA attributes	1310

Keyboard interaction	1310
Ensuring accessibility	1311
See also	1311
Internationalization in ##Platform_Name## Chart control.....	1311
Localization in ##Platform_Name## Chart control.....	1313
Ej1 api migration in ##Platform_Name## Chart control	1315
Annotations.....	1315
Columns	1316
CommonSeriesOptions	1317
Crosshair	1317
3D chart.....	1430
Canvas rendering	1430
Indicators	1430
Legend.....	1434
primaryXAxis	1436
primaryYAxis	1615
Axes.....	1624
Rows.....	1737
Series.....	1737
marker.....	1744
TrendLines.....	1748
StripLines.....	1750
Multilevel Labels	1751
Methods.....	1753
Events.....	1753
How To	1760
Live chart in ##Platform_Name## Chart control	1760
Prevent data label in ##Platform_Name## Chart control	1762
Tool tip format in ##Platform_Name## Chart control	1764
Add series in ##Platform_Name## Chart control	1766
Points customization in ##Platform_Name## Chart control	1767
Stacking total in ##Platform_Name## Chart control.....	1770
Selected data grid in ##Platform_Name## Chart control.....	1772
Marker customization in ##Platform_Name## Chart control	1774
Legend customization in ##Platform_Name## Chart control	1776

Tool tip table in ##Platform_Name## Chart control	1778
Footer in ##Platform_Name## Chart control	1779
Threshold in ##Platform_Name## Chart control	1781
Grid data chart in ##Platform_Name## Chart control	1783
Percentage tool tip in ##Platform_Name## Chart control	1785
Data label template in ##Platform_Name## Chart control	1786
Hide tool tip in ##Platform_Name## Chart control	1788
Dotted line in ##Platform_Name## Chart control	1790
Grid data pie in ##Platform_Name## Chart control	1791
Down sampling in ##Platform_Name## Chart control	1793
Clicked data in ##Platform_Name## Chart control	1810
Initial scrollbar in ##Platform_Name## Chart control	1812
Legend in table in ##Platform_Name## Chart control	1814
Syn pan in ##Platform_Name## Chart control	1818
Axis hide in ##Platform_Name## Chart control	1819
Exact date in ##Platform_Name## Chart control	1821
Dialog chart in ##Platform_Name## Chart control	1822
Series visible in ##Platform_Name## Chart control	1823
Dynamic chart in ##Platform_Name## Chart control	1824
CheckBox	1825
Label and size in ##Platform_Name## Check box control	1825
Label	1825
Size	1826
See Also	1828
Accessibility in ##Platform_Name## Check box control	1828
WAI-ARIA attributes	1829
Keyboard interaction	1829
Ensuring accessibility	1829
See also	1829
How To	1830
Customized checkbox in ##Platform_Name## Check box control	1830
Name and value in form submit in ##Platform_Name## Check box control	1837
Right to left in ##Platform_Name## Check box control	1839
Chips	1840
Types in ##Platform_Name## Chips control	1840

Input Chip.....	1840
Choice Chip	1841
Filter Chip	1842
Action Chip.....	1843
Customization in ##Platform_Name## Chips control	1846
Styles	1846
Leading Icon	1848
Avatar.....	1849
Avatar Content.....	1851
Trailing Icon.....	1852
Outline Chip	1854
Style in ##Platform_Name## Chips control.....	1855
Customizing the chip text	1855
Customizing the chip icon	1855
Customizing the chip delete button.....	1856
Customizing the chip outline	1856
Customizing the chip on selection	1856
Customizing the chip avatar text	1857
Accessibility in ##Platform_Name## Chips component	1857
WAI-ARIA attributes.....	1858
Keyboard interaction	1858
Ensuring accessibility	1858
See also	1859
Circular Gauge.....	1859
Gauge dimensions in ##Platform_Name## Circular gauge control.....	1859
Size for Container.....	1859
Size for Circular Gauge.....	1860
Gauge axes in ##Platform_Name## Circular gauge control	1861
Axis Customization.....	1861
Angles and Direction	1862
Axis Radius	1864
Ticks.....	1866
Labels	1868
Minimum and Maximum	1876
Multiple Axes	1876

Gauge ranges in ##Platform_Name## Circular gauge control	1878
Start and End.....	1878
Customization	1879
Radius.....	1879
Dragging Range	1881
Multiple Ranges	1882
Rounded corner radius	1884
Gradient Color.....	1885
See also	1890
Gauge pointers in ##Platform_Name## Circular gauge control.....	1890
Needle Pointers.....	1891
RangeBar Pointer	1894
Rounded corner for range bar pointer	1896
Marker Pointer.....	1897
Dragging pointer	1899
Multiple Pointers	1900
Animation.....	1902
Gradient Color.....	1903
Gauge annotations in ##Platform_Name## Circular gauge control.....	1906
Content	1907
Position	1908
Sub Gauge	1909
See also	1912
Animation in ##Platform_Name## Circular Gauge control.....	1912
Gauge legend in ##Platform_Name## Circular gauge control	1915
Legend customization	1915
Toggle option in legend	1918
Paging support in legend	1919
Legend text customization.....	1920
Gauge user interaction in ##Platform_Name## Circular gauge control	1922
Tooltip for pointers	1922
Tooltip for ranges.....	1924
Pointer Drag	1926
Gauge appearance in ##Platform_Name## Circular gauge control	1928
Gauge Title	1928

Gauge Position	1928
Area Customization.....	1930
Radius calculation based on angles	1933
Accessibility in ##Platform_Name## Circular gauge control.....	1934
WAI-ARIA attributes.....	1934
Screen reading in Circular Gauge	1935
Ensuring accessibility	1935
See also	1935
Internationalization in ##Platform_Name## Circular gauge control.....	1935
Globalization	1935
Right-to-left.....	1936
Ej1 api migration in ##Platform_Name## Circular gauge control	1938
Circular gauge dimensions	1938
Axis Line	1939
Ticks.....	1939
Labels	1941
Ranges	1942
Needle Pointer	1943
Marker Pointer.....	1944
Rangebar Pointer	1945
Annotations.....	1945
Appearance	1946
Events.....	1947
How To	1948
Gauge range in ##Platform_Name## Circular gauge control.....	1948
Color Picker	1950
Mode and value in ##Platform_Name## Color picker control	1950
Rendering palette at initial load	1950
Color value	1952
See Also	1953
Localization in ##Platform_Name## Color picker control	1954
Localization	1954
Right to Left - RTL.....	1955
See Also	1957
Accessibility in ##Platform_Name## Color picker control	1957

WAI-ARIA attributes.....	1958
Keyboard interaction	1958
Ensuring accessibility	1959
See also	1959
Style and appearance in ##Platform_Name## Color picker control	1959
How To	1959
Hide control buttons in ##Platform_Name## Color picker control.....	1959
Render palette alone in ##Platform_Name## Color picker control	1961
Colorpicker in dropdownbutton in ##Platform_Name## Color picker control	1962
Customize colorpicker in ##Platform_Name## Color picker control	1965
Handle no color support in ##Platform_Name## Color picker control.....	1978
Disabled in ##Platform_Name## Color picker control	1984
ComboBox.....	1985
Tags in ##Platform_Name## Combo box control.....	1985
Select element	1985
UL element.....	1987
Input element	1988
Data binding in ##Platform_Name## Combo box control.....	1988
Binding local data.....	1988
Binding remote data	1991
See Also	1993
Templates in ##Platform_Name## Combo box control	1993
Item template	1993
Group template.....	1994
Header template	1995
Footer template	1997
No records template	1998
Action failure template	1999
See Also	2000
Grouping in ##Platform_Name## Combo box control	2000
HTML select.....	2001
Customization	2003
See Also	2003
Filtering in ##Platform_Name## Combo box control.....	2003
Limit the minimum filter character.....	2004

Change the filter type	2006
Case sensitive filtering	2007
Diacritics Filtering.....	2009
See Also	2010
Virtualization in ComboBox Component	2010
Binding local data.....	2011
Binding Remote data.....	2012
Grouping with Virtualization.....	2013
Filtering with Virtualization.....	2015
Localization in ##Platform_Name## Combo box control	2016
Loading translations.....	2017
See Also	2018
Style in ##Platform_Name## Combo box control	2018
Customizing the appearance of wrapper element	2018
Customizing the dropdown icon's color	2018
Customizing the focus color.....	2019
Customizing the outline theme's focus color	2019
Customizing the disabled component's text color	2019
Customizing the float label element's focusing color	2019
Customizing the color of the placeholder text	2020
Customizing the text selection color.....	2020
Customizing the background color of focus, hover, and active item's	2020
Customizing the appearance of pop-up element	2021
Adding mandatory asterisk to placeholder and float label.....	2021
Accessibility in ##Platform_Name## Combo box control	2022
WAI-ARIA attributes.....	2023
Keyboard interaction	2023
Ensuring accessibility	2025
See also	2025
How To	2025
Autofill in ##Platform_Name## Combo box control	2025
Cascading in ##Platform_Name## Combo box control.....	2027
Icons support in ##Platform_Name## Combo box control	2029
Achieve virtual scrolling in ##Platform_Name## Combo box control.....	2030
ContextMenu	2032

Icons and navigation in ##Platform_Name## Context menu control	2032
Icons	2032
Navigation	2034
See Also	2035
Template and multilevel nesting in ##Platform_Name## Context menu control.....	2035
Template	2035
Multilevel nesting	2037
See Also	2038
Accessibility in ##Platform_Name## Context menu control	2038
WAI-ARIA attributes.....	2039
Keyboard interaction	2039
Ensuring accessibility	2040
See also	2040
Style and appearance in ##Platform_Name## Context menu control.....	2040
How To	2040
Populate menu items with data source in ##Platform_Name## Context menu control	2040
Open and close contextmenu in ##Platform_Name## Context menu control	2042
Change menu items dynamically in ##Platform_Name## Context menu control	2043
Template in ##Platform_Name## Context menu control	2045
Underline a character in the item text in ##Platform_Name## Context menu control.....	2050
Open a dialog on contextmenu item click in ##Platform_Name## Context menu control	2052
Change animation settings in ##Platform_Name## Context menu control.....	2054
Add or remove context menu items in ##Platform_Name## Context menu control	2055
Enable or disable context menu items in ##Platform_Name## Context menu control.....	2056
DashboardLayout	2057
Setting size of cells in ##Platform_Name## Dashboard layout control	2057
Modifying cell size.....	2057
Setting cell spacing.....	2059
Graphical representation of layout.....	2060
Rendering component in right-to-left direction	2061
Panels	2063
Position sizing of panels in ##Platform_Name## Dashboard layout control	2063
Setting header of panels in ##Platform_Name## Dashboard layout control	2066
Add remove panels in ##Platform_Name## Dashboard layout control.....	2070
Interaction With Panels	2073

Dragging moving of panels in ##Platform_Name## Dashboard layout control	2073
Moving panels in ##Platform_Name## Dashboard layout control	2079
Resizing of panels in ##Platform_Name## Dashboard layout control	2080
Floating of panels in ##Platform_Name## Dashboard layout control	2084
Responsive adaptive in ##Platform_Name## Dashboard layout control.....	2085
Save restore in ##Platform_Name## Dashboard layout control.....	2087
Style in ##Platform_Name## Dashboard layout control	2089
Customizing the dashboard layout panel header	2089
Customizing the dashboard layout panel content.....	2089
Customizing the dashboard layout panel resize icon	2089
Customizing the dashboard layout panel background	2089
How To	2090
Resize the panel dynamically in ##Platform_Name## Dashboard layout control	2090
Accessibility in ##Platform_Name## Dashboard Layout component	2091
WAI-ARIA attributes.....	2092
Keyboard interaction	2093
Ensuring accessibility	2093
See also	2093
DataManager	2093
Data binding in ##Platform_Name## Data control	2093
Local data binding	2093
Remote data binding.....	2094
See Also	2096
Adaptors in ##Platform_Name## Data control	2096
Json adaptor	2096
Url adaptor	2098
OData adaptor	2098
ODataV4 adaptor	2100
Web API adaptor	2101
WebMethod Adaptor.....	2102
Custom Data Adaptor	2103
GraphQL Adaptor	2105
Writing custom adaptor.....	2110
Querying in ##Platform_Name## Data control	2112
Specifying resource name using `from`	2112

Projection using `select`	2113
Eager loading navigation properties	2115
Sorting	2116
Filtering	2118
Searching	2121
Grouping	2122
Paging	2124
Aggregation	2125
Hierarchical query	2127
Manipulation in ##Platform_Name## Data control	2129
Insert	2129
Update	2131
Remove	2133
Batch Edit Operation	2135
State persistence in ##Platform_Name## Data control	2138
Preventing a query from persistence	2138
How to get or set the existing persisted data	2139
Restoring the initial state of Datamanager	2141
Use case example demonstrating state persistence with the DataManager	2141
How to in ##Platform_Name## Data control	2142
Work in offline mode	2142
Sending additional parameters to server	2144
Adding custom headers	2145
DatePicker	2145
Date range in ##Platform_Name## Datepicker control	2145
Date format in ##Platform_Name## Datepicker control	2147
Date masking in ##Platform_Name## Datepicker control	2148
Configure Mask Placeholder	2150
Globalization in ##Platform_Name## Datepicker control	2152
Right-To-Left	2156
Strict mode in ##Platform_Name## Datepicker control	2159
Customization in ##Platform_Name## Datepicker control	2162
Adding mandatory asterisk to placeholder and float label	2163
See Also	2165
Date views in ##Platform_Name## Datepicker control	2165

Start view	2165
Depth view	2166
Accessibility in ##Platform_Name## Datepicker control	2167
WAI-ARIA attributes.....	2168
Keyboard Interaction	2168
Ensuring accessibility	2170
See also	2170
Style appearance in ##Platform_Name## Datepicker control	2171
Customizing the appearance of DatePicker wrapper element.....	2171
Customizing the DatePicker icon element.....	2171
Customizing the Calendar popup of the DatePicker.....	2171
Full screen mode support in mobiles and tablets.....	2171
How To	2174
Disabled the datepicker component in ##Platform_Name## Datepicker control	2174
Set the placeholder in ##Platform_Name## Datepicker control	2175
Set the readonly in ##Platform_Name## Datepicker control	2176
Prevent the popup close in ##Platform_Name## Datepicker control.....	2177
Customize the datepicker day header in ##Platform_Name## Datepicker control.....	2178
Open datepicker popup on input click in ##Platform_Name## Datepicker control	2179
Client side validation in ##Platform_Name## Datepicker control	2180
DateRangePicker	2182
Range restriction in ##Platform_Name## Daterangepicker control	2182
Restrict the range within a range.....	2182
Range span	2183
Strict mode.....	2184
Globalization in ##Platform_Name## Daterangepicker control	2185
Right-To-Left	2190
Customize the date format	2192
Customization in ##Platform_Name## Daterangepicker control	2193
Day cell format.....	2193
Preset Ranges.....	2194
First day of week	2195
See Also	2196
Accessibility in ##Platform_Name## Daterangepicker control	2196
WAI-ARIA attributes.....	2197

Keyboard Interaction	2198
Ensuring accessibility	2200
See also	2200
Style appearance in ##Platform_Name## Daterangepicker control	2200
Customizing the appearance of DateRangePicker wrapper element	2200
Customizing the DateRangePicker icon element	2200
Customizing the DateRangePicker popup calendar header	2200
Customizing the DateRangePicker popup calendar header title	2201
Customizing the DateRangePicker popup calendar content	2201
Customizing the DateRangePicker popup calendar content title	2201
Customizing the DateRangePicker popup calendar previous and next icon	2201
Customizing the DateRangePicker popup calendar date cell grid on hovering	2202
Customizing the DateRangePicker popup calendar primary button in footer	2202
Customizing the DateRangePicker popup calendar cancel button in footer	2202
Customizing the footer element in the DateRangePicker popup calendar	2203
Customizing the selected date cell grid in the DateRangePicker popup calendar	2203
Full screen mode support in mobiles and tablets	2203
How To	2206
Disable the daterangepicker component in ##Platform_Name## Daterangepicker control	2206
Set the placeholder in ##Platform_Name## Daterangepicker control	2207
Customization using cssclass in ##Platform_Name## Daterangepicker control	2208
Customize the daterangepicker day header in ##Platform_Name## Daterangepicker control	2209
DateTimePicker	2211
Globalization in ##Platform_Name## Datetimepicker control	2211
Right-To-Left	2215
Strict mode in ##Platform_Name## Datetimepicker control	2217
Date time range in ##Platform_Name## Datetimepicker control	2220
Date time format in ##Platform_Name## Datetimepicker control	2221
Date time masking in ##Platform_Name## Datetimepicker control	2222
Configure Mask Placeholder	2225
Customization in ##Platform_Name## Datetimepicker control	2226
Day and Time Cell format	2226
Adding mandatory asterisk to placeholder and float label	2227
See Also	2228
Accessibility in ##Platform_Name## Datetimepicker control	2228

WAI-ARIA attributes.....	2229
Keyboard Interaction	2230
Ensuring accessibility	2232
See also	2232
Style appearance in ##Platform_Name## Datetimepicker control.....	2232
Customizing the appearance of DateTimePicker wrapper element.....	2232
Customizing the DateTimePicker icons element	2233
Customizing the time picker popup in the DateTimePicker	2233
Customizing the Calendar popup of the DateTimePicker	2233
Full screen mode support in mobiles and tablets.....	2233
How To	2236
Disable the datetimepicker component in ##Platform_Name## Datetimepicker control	2236
Set the placeholder in ##Platform_Name## Datetimepicker control	2237
Customize the datetimepicker day header in ##Platform_Name## Datetimepicker control	2238
Diagram.....	2239
Getting started in ##Platform_Name## Diagram control	2239
Dependencies.....	2239
Installation and Configuration	2240
Add diagram to the project.....	2241
Module Injection.....	2243
Flow Diagram	2243
Automatic organization chart	2247
Shapes in ##Platform_Name## Diagram control	2251
Text	2251
Image.....	2252
Image alignment	2256
HTML.....	2258
HTML Node With Template	2259
Native.....	2260
SVG content alignment	2264
Basic shapes	2265
Path	2267
Flow Shapes	2268
Bpmn shapes in ##Platform_Name## Diagram control	2270
Event	2272

Gateway	2275
Activity	2277
Tasks.....	2278
Subprocess	2281
Event subprocess	2282
Transaction subprocess.....	2284
Process	2285
Loop.....	2285
Compensation	2287
Call.....	2289
Adhoc	2290
Boundary.....	2291
Data	2293
Datasource	2295
Artifact	2296
Text annotation.....	2296
Group	2298
BPMN flows.....	2299
Association	2299
Sequence.....	2300
Message	2302
UML diagram in ##Platform_Name## Diagram control	2303
UML Class Diagram	2303
UML Class Diagram Shapes	2304
UML Class Relationships	2309
How to add UML child at runtime.....	2318
Adding UML Nodes in Symbol palette	2319
Editing in UML nodes	2321
UML Activity diagram.....	2322
UML Activity diagram Shapes	2322
Group in ##Platform_Name## Diagram control.....	2327
Create group	2327
Add group when initializing diagram	2327
Add group at runtime	2331
Add children To group at runtime	2332

Remove children from group at runtime	2332
Container.....	2334
Difference between a basic group and containers	2338
Interaction.....	2338
See Also	2338
Swim lane in ##Platform_Name## Diagram control	2338
Create a swimlane.....	2338
Lanes	2345
Phase	2362
Add swimlane to palette	2369
Limitations.....	2372
Ports in ##Platform_Name## Diagram control	2372
Create port.....	2373
Add ports when initializing nodes.....	2373
Add ports at runtime.....	2375
Remove ports at runtime	2377
Update port at runtime.....	2379
Appearance	2380
Offset.....	2382
Constraints	2382
Grid lines in ##Platform_Name## Diagram control.....	2382
Customize the gridlines visibility.....	2382
Appearance	2383
Line intervals.....	2385
Snapping.....	2386
Snap to lines.....	2386
Customization of snap intervals.....	2387
Snap to objects.....	2389
Page settings in ##Platform_Name## Diagram control.....	2390
Page size and appearance.....	2391
Set background image.....	2393
Multiple page and page breaks.....	2395
Boundary constraints.....	2397
Scroll settings in ##Platform_Name## Diagram control.....	2400
Get current scroll status.....	2400

Define scroll status.....	2400
Update scroll status	2401
AutoScroll.....	2402
Autoscroll border	2404
Scroll limit	2405
Scroll Padding.....	2407
Scrollable Area	2408
UpdateViewport.....	2410
Accessibility in ##Platform_Name## Diagram control	2410
WAI-ARIA attributes.....	2411
Aria-label	2411
Keyboard interaction	2412
Ensuring accessibility	2412
See also	2412
Tool tip in ##Platform_Name## Diagram control.....	2412
Default tooltip.....	2413
Common tooltip for all nodes and connectors	2413
Tooltip for a specific node/connector.....	2415
Tooltip for Ports	2417
Tooltip template content.....	2419
Tooltip alignments	2421
Tooltip animation.....	2425
User handle in ##Platform_Name## Diagram control	2427
Alignment.....	2427
Fixed user handles	2429
Initialization an fixed user handles	2430
Customization	2431
Customizing the node fixed user handle	2433
Customizing the connector fixed user handle	2436
Style in ##Platform_Name## Diagram control	2441
Customizing the connector end point handle.....	2441
Customizing the connector end point handle when connected.....	2441
Customizing the connector end point handle when disabled	2441
Customizing the bezier connector handle	2441
Customizing the bezier connector line	2442

Customizing the resize handle	2442
Customizing the selector pivot line.....	2442
Customizing the selector border.....	2442
Customizing the rotate handle	2442
Customizing the symbolpalette while hovering	2443
Customizing the symbolpalette when selected	2443
Customizing the ruler.....	2443
Customizing the ruler overlap.....	2443
Customizing the text edit.....	2444
Customizing the text edit on selection	2444
Ruler in ##Platform_Name## Diagram control	2444
Adding Rulers to the Diagram.....	2444
Customizing the Ruler	2445
Context menu in ##Platform_Name## Diagram control	2447
Customize context menu	2447
Template Support for Context menu.....	2452
Context menu events.....	2455
Overview in ##Platform_Name## Diagram control.....	2458
Create overview	2458
Dialog	2461
Template in ##Platform_Name## Dialog control	2461
Header.....	2461
Footer.....	2461
Content	2461
See Also	2463
Animation in ##Platform_Name## Dialog control	2463
Resize in ##Platform_Name## Dialog control	2466
Dialog utility in ##Platform_Name## Dialog control	2467
Alert dialog.....	2468
Confirm dialog.....	2470
Close utility dialog.....	2472
Style in ##Platform_Name## Dialog control	2474
Customizing the dialog header	2474
Customizing the dialog content	2474
Customizing modal dialog overlay	2475

Customizing the dialog resize icon.....	2475
Customizing the dialog close button.....	2475
Customizing the dialog footer button.....	2475
Localization in ##Platform_Name## Dialog control	2476
Loading translations.....	2476
Accessibility in ##Platform_Name## Dialog control.....	2477
WAI-ARIA attributes.....	2478
Keyboard interaction	2479
See Also	2481
Ensuring accessibility	2481
See also	2481
How To	2481
Create nested dialog in ##Platform_Name## Dialog control	2481
Position the dialog on center of the page on scrolling in ##Platform_Name## Dialog control	2483
Load dialog content using ajax in ##Platform_Name## Dialog control.....	2485
Render a dialog without header in ##Platform_Name## Dialog control	2485
Show dialog with full screen in ##Platform_Name## Dialog control	2487
Display a dialog with custom position in ##Platform_Name## Dialog control	2489
Prevent closing of modal dialog in ##Platform_Name## Dialog control.....	2490
Prevent the focus on the first element in ##Platform_Name## Dialog control.....	2493
Prevent opening of the dialog in ##Platform_Name## Dialog control	2495
Read all the values from dialog on button click in ##Platform_Name## Dialog control.....	2498
Customize the dialog appearance in ##Platform_Name## Dialog control.....	2501
Close dialog while click on outside of dialog in ##Platform_Name## Dialog control	2502
Add an icons to dialog buttons in ##Platform_Name## Dialog control	2504
Add a minimize maximize buttons in ##Platform_Name## Dialog control.....	2506
Setting max height to the dialog in ##Platform_Name## Dialog control.....	2509
DocumentEditor.....	2511
Overview	2511
Key Features.....	2511
Supported Web platforms	2512
Feature module in ##Platform_Name## Document editor control	2513
Import in ##Platform_Name## Document editor control.....	2515
Import document from local machine	2516
Convert word documents into SFDT	2518

Compatibility with Microsoft Word	2520
See Also	2521
Export in ##Platform_Name## Document editor control	2522
SFDT export	2522
Word export	2523
Text export	2524
Export as blob	2525
See Also	2528
Server Deployment	2528
Word processor server docker image overview in ##Platform_Name## Document editor control	2528
Provide your license key for activation	2529
Provide your license key for activation	2530
Provide your license key for activation	2531
Provide your license key for activation	2532
How to deploy word processor server docker container in azure app service in ##Platform_Name## Document editor control	2533
How to deploy word processor server docker container in azure kubernetes service in ##Platform_Name## Document editor control	2534
How to publish documenteditor web api application in azure app service from visual studio in ##Platform_Name## Document editor control	2537
How to deploy documenteditor java web api in azure in ##Platform_Name## Document editor control	2539
Accessibility in Angular Document editor component	2541
Keyboard interaction	2542
Ensuring accessibility	2542
See also	2543
Image in ##Platform_Name## Document editor control	2543
Alternate text	2545
Image resizing	2545
Changing size	2545
Text wrapping style	2546
Positioning the image	2546
See Also	2546
Shapes in ##Platform_Name## Document editor control	2546
Supported shapes	2546

Text box Shape	2546
Text wrapping style	2547
Positioning the shape.....	2547
Text wrapping style in ##Platform_Name## Document editor control	2547
In-Line with Text.....	2547
In Front of Text.....	2547
Top and Bottom	2548
Behind	2548
Square	2548
Bookmark in ##Platform_Name## Document editor control	2549
Add bookmark.....	2549
Select Bookmark	2549
Delete Bookmark	2549
Get Bookmark from document.....	2550
Get Bookmark from selection	2550
Replace bookmark content	2550
Show or Hide bookmark.....	2550
Bookmark Dialog	2550
See Also	2552
Link in ##Platform_Name## Document editor control	2552
Navigate a hyperlink	2552
Copy link.....	2554
Add hyperlink	2554
Customize screen tip.....	2555
Remove hyperlink	2556
Hyperlink dialog	2556
See Also	2558
Table in ##Platform_Name## Document editor control	2558
Create a table.....	2558
Insert rows	2558
Delete table.....	2559
Delete row.....	2559
Delete column.....	2560
Merge cells.....	2560
Positioning the table	2560

How to work with tables	2560
See Also	2563
Table of contents in ##Platform_Name## Document editor control.....	2563
Inserting table of contents.....	2563
Update or edit table of contents	2566
See Also	2567
Header footer in ##Platform_Name## Document editor control	2567
Go to header footer region	2568
Header and footer distance	2568
Close header footer region	2568
Link to previous.....	2568
See Also	2569
Text format in ##Platform_Name## Document editor control	2569
Bold	2569
Italic.....	2569
Underline property	2570
Strikethrough property	2570
Superscript property	2571
Subscript property	2571
Size	2571
Color.....	2572
Font	2572
Highlight color.....	2572
Toolbar with options for text formatting.....	2572
See Also	2576
Paragraph format in ##Platform_Name## Document editor control	2576
Indentation.....	2576
Special indentation	2577
Increase indent	2577
Decrease indent	2577
Text alignment	2577
Line spacing and its type.....	2578
Paragraph spacing.....	2578
Pagination properties.....	2579
Paragraph Border	2579

Show or Hide Paragraph marks.....	2580
Toolbar with paragraph formatting options	2580
See Also	2584
List format in ##Platform_Name## Document editor control	2584
Create bullet list.....	2584
Create numbered list	2584
Clear list.....	2585
Working with lists	2585
Editing numbered list.....	2587
See Also	2587
Table format in ##Platform_Name## Document editor control	2587
Cell margins.....	2587
Background color	2588
Cell spacing.....	2588
Cell vertical alignment.....	2588
Table alignment	2589
Cell width	2589
Table width	2589
Apply borders.....	2590
Working with row formatting	2591
See Also	2592
Section format in ##Platform_Name## Document editor control	2592
Page size.....	2592
Page margins.....	2593
Header distance	2593
Footer distance	2593
Columns	2593
Breaks.....	2594
See Also	2594
Comments in ##Platform_Name## Document editor control	2594
Add a new comment.....	2594
Comment navigation.....	2594
Delete comment	2595
Delete all comment.....	2595
Protect the document in comments only mode.....	2595

Fields in ##Platform_Name## Document editor control.....	2596
Adding Fields	2596
Update fields	2596
Get field info	2597
Set field info	2597
See Also	2598
Form fields in ##Platform_Name## Document editor control.....	2598
Insert form field	2598
Get form field names	2598
Get form field properties	2598
Set form field properties	2599
Export form field data	2600
Import form field data	2600
Reset form fields	2600
Protect the document in form filling mode	2600
Clipboard in ##Platform_Name## Document editor control	2601
Copy	2601
Cut	2601
Paste.....	2601
Local paste (copy/paste within control)	2601
Paste with formatting	2603
See Also	2603
Collaborative Editing (preview).....	2604
Prerequisites	2604
How to enable collaborative editing in client side.....	2604
How to enable collaborative editing in ASP.NET Core.....	2606
History in ##Platform_Name## Document editor control	2612
Enable or disable history.....	2612
Undo and redo	2613
Stack size	2613
See Also	2613
Find and replace in ##Platform_Name## Document editor control	2613
Options pane.....	2614
Search.....	2615
Search results	2616

SearchResultsChange event.....	2617
Customize find and replace.....	2618
Keyboard shortcut in ##Platform_Name## Document editor control	2620
Text formatting	2620
Paragraph formatting.....	2620
Clipboard	2620
Keyboard shortcut to navigate around the document	2621
Keyboard shortcut to extend selection.....	2621
Find and Replace	2622
Create, Save and Print document	2622
Edit Operation.....	2622
Insert special characters	2622
Dialog	2622
See Also	2622
Scrolling zooming in ##Platform_Name## Document editor control	2623
Zooming	2627
Page Fit Type	2628
Zoom option using UI.....	2628
Print in ##Platform_Name## Document editor control	2632
Improve print quality	2634
Print using window object	2635
Page setup.....	2635
See Also	2637
Dialog in ##Platform_Name## Document editor control.....	2637
Font Dialog	2637
Paragraph dialog	2639
Table dialog	2640
Bookmark dialog	2641
Hyperlink dialog	2642
Table of contents dialog.....	2644
Styles Dialog	2645
Style dialog.....	2646
List dialog	2647
Borders and shading dialog.....	2649
Table options dialog.....	2650

Table properties dialog	2651
Page setup dialog	2653
Column dialog	2654
See Also	2655
R t l in ##Platform_Name## Document editor control	2655
Chart in ##Platform_Name## Document editor control	2662
Supported Chart Types	2682
Content control in ##Platform_Name## Document editor control	2682
Types of Content Controls	2683
Restrict editing in ##Platform_Name## Document editor control	2683
Set current user	2683
Highlighting the text area	2683
Restrict Editing Pane	2683
See Also	2690
Spell check in ##Platform_Name## Document editor control	2690
Features	2691
Enable SpellCheck	2691
Disable SpellCheck	2691
Spell check settings	2691
Context menu	2693
Global local in ##Platform_Name## Document editor control	2695
Localization	2695
Document Editor	2695
Color Picker	2700
Notes in ##Platform_Name## Document editor control	2700
Insert footnotes	2701
Insert endnotes	2701
Update or edit footnotes and endnotes	2701
How To	2702
Override the keyboard shortcuts in ##Platform_Name## Document editor control	2702
Customize context menu in ##Platform_Name## Document editor control	2705
Customize tool bar in ##Platform_Name## Document editor control	2710
Change document view in ##Platform_Name## Document editor control	2711
Open default document in ##Platform_Name## Document editor control	2711
Read only default in ##Platform_Name## Document editor control	2717

Open document by address in ##Platform_Name## Document editor control	2722
Deploy document editor component for mobile in ##Platform_Name## Document editor control	2724
Disable optimized text measuring in ##Platform_Name## Document editor control	2725
Get the selected content in ##Platform_Name## Document editor control	2726
Set default format in document editor in ##Platform_Name## Document editor control	2728
Show hide spinner in ##Platform_Name## Document editor control	2730
Resize document editor in ##Platform_Name## Document editor control	2733
Export document as pdf in ##Platform_Name## Document editor control	2735
Customize font family drop down in ##Platform_Name## Document editor control	2739
Auto save document in document editor in ##Platform_Name## Document editor control	2740
Retrieve the bookmark content as text in ##Platform_Name## Document editor control	2743
Get current word in ##Platform_Name## Document editor control	2746
Insert page number and navigate to page in ##Platform_Name## Document editor control	2747
Move selection to specific position in ##Platform_Name## Document editor control	2749
Disable header and footer edit in document editor in ##Platform_Name## Document editor control	2750
Insert text in current position in ##Platform_Name## Document editor control	2752
Change the cursor color in document editor in ##Platform_Name## Document editor control ..	2755
Hide tool bar and properties pane in ##Platform_Name## Document editor control	2756
Insert text or image in table programmatically in ##Platform_Name## Document editor control	2757
Change the default search highlight color in ##Platform_Name## Document editor control	2759
Disable auto focus in ##Platform_Name## Document editor control	2760
Disable drag and drop in ##Platform_Name## Document editor control	2760
Enable ruler	2761
DropDown Menu	2763
Icons in ##Platform_Name## Drop down button control	2763
DropDownButton icons	2763
Vertical button	2769
See Also	2771
Popup items in ##Platform_Name## Drop down button control	2772
Icons	2772
Navigations	2774
Template	2777
Separator	2783

See Also	2787
Accessibility in ##Platform_Name## Drop down button control.....	2787
WAI-ARIA attributes.....	2788
Keyboard interaction	2788
Ensuring accessibility	2789
See also	2789
How To	2789
Change caret icon in ##Platform_Name## Drop down button control.....	2789
Create dropdownbutton with rounded corner in ##Platform_Name## Drop down button control	2791
Create right to left dropdownbutton in ##Platform_Name## Drop down button control.....	2792
Customize icon and width in ##Platform_Name## Drop down button control.....	2794
Disable a dropdownbutton in ##Platform_Name## Drop down button control	2796
Group popup items with listview component in ##Platform_Name## Drop down button control	2798
Hide dropdown arrow in ##Platform_Name## Drop down button control.....	2800
Open a dialog on popup item click in ##Platform_Name## Drop down button control	2802
Position popup open in ##Platform_Name## Drop down button control.....	2804
Underline a character in the item text in ##Platform_Name## Drop down button control.....	2806
DropDownList	2808
Tags in ##Platform_Name## Drop down list control.....	2808
Select element	2808
UL element.....	2809
Input element	2810
Data binding in ##Platform_Name## Drop down list control	2810
Binding local data.....	2810
Binding remote data	2814
See Also	2815
Templates in ##Platform_Name## Drop down list control.....	2815
Item template	2815
Value template.....	2817
Group template.....	2818
Header template	2819
Footer template	2820
No records template	2822

Action failure template	2823
See Also	2824
Grouping in ##Platform_Name## Drop down list control.....	2824
HTML select.....	2825
Customization	2827
See Also	2827
Filtering in ##Platform_Name## Drop down list control.....	2827
Limit the minimum filter character	2828
Change the filter type	2830
Case sensitive filtering	2831
Diacritics Filtering.....	2833
See Also	2834
Virtualization in DropDown List	2834
Binding local data.....	2835
Binding Remote data.....	2836
Grouping with Virtualization.....	2837
Filtering with Virtualization.....	2839
Localization in ##Platform_Name## Drop down list control.....	2840
Loading translations.....	2841
See Also	2842
Style in ##Platform_Name## Drop down list control.....	2842
Customizing the appearance of wrapper element	2842
Customizing the dropdown icon's color	2842
Customizing the focus color	2843
Customizing the outline theme's focus color	2843
Customizing the disabled component's text color	2843
Customizing the float label element's focusing color	2843
Customizing the color of the placeholder text	2844
Customizing the background color of focus, hover, and active item's	2844
Customizing the appearance of pop-up element	2844
Adding mandatory asterisk to placeholder and float label.....	2844
Accessibility in ##Platform_Name## Drop down list control	2845
WAI-ARIA attributes.....	2846
Keyboard interaction	2847
Ensuring accessibility	2849

See also	2849
How To	2849
Add item in ##Platform_Name## Drop down list control	2849
Cascading in ##Platform_Name## Drop down list control.....	2851
Clear item in ##Platform_Name## Drop down list control	2853
Close popup in ##Platform_Name## Drop down list control.....	2854
Group header in ##Platform_Name## Drop down list control	2855
Highlight filtering in ##Platform_Name## Drop down list control.....	2857
Icons support in ##Platform_Name## Drop down list control.....	2858
Incremental search in ##Platform_Name## Drop down list control.....	2859
Modify data in ##Platform_Name## Drop down list control	2859
Multiple cascading in ##Platform_Name## Drop down list control.....	2861
Remote data bind in ##Platform_Name## Drop down list control.....	2863
Remove item in ##Platform_Name## Drop down list control	2865
Search on filtering in ##Platform_Name## Drop down list control	2866
Tooltip in ##Platform_Name## Drop down list control	2868
Value change in ##Platform_Name## Drop down list control	2869
Value support in ##Platform_Name## Drop down list control	2871
Achieve virtual scrolling in ##Platform_Name## Drop down list control	2871
Dropdown Tree	2873
Data binding in ##Platform_Name## Drop down tree control	2873
Local data	2873
Remote data.....	2877
Prevent Node selection.....	2879
Templates in ##Platform_Name## Drop down tree control.....	2880
Item template	2880
Header template	2882
Footer template	2883
No records template	2885
Action failure template	2886
Custom template to show selected items in input.....	2887
Checkbox in ##Platform_Name## Drop down tree control	2890
Auto Check	2892
Select All.....	2894
Localization in ##Platform_Name## Drop down tree control.....	2895

Accessibility in ##Platform_Name## Dropdown Tree component	2895
WAI-ARIA attributes.....	2896
Keyboard interaction	2897
Ensuring accessibility	2898
See also	2898
FileManager	2898
User interface in ##Platform_Name## File manager control.....	2898
Toolbar	2899
Files and folders navigation	2900
View	2901
Context menu.....	2902
File operations in ##Platform_Name## File manager control.....	2902
Folder Upload support	2903
File operation request and response Parameters	2908
File request and response contents.....	2924
Action Buttons	2925
Views in ##Platform_Name## File manager control	2926
Largelcons View	2927
Details View.....	2928
Customization in ##Platform_Name## File manager control	2929
Context menu customization.....	2929
Details view customization	2931
Navigation pane customization	2932
Show/Hide file extension	2934
Show/Hide hidden items.....	2935
Show/Hide thumbnail images in large icons view	2936
Toolbar customization	2938
Upload customization	2939
Tooltip customization	2940
Multiple selection in ##Platform_Name## File manager control	2943
Drag and drop in ##Platform_Name## File manager control	2945
File system provider in ##Platform_Name## File manager control	2946
ASP.NET Core file system provider	2947
ASP.NET MVC 5 file system provider	2948
ASP.NET Core Azure cloud file system provider	2948

ASP.NET MVC Azure cloud file system provider	2950
ASP.NET Core Amazon S3 cloud file provider	2951
ASP.NET MVC Amazon S3 cloud file provider	2952
File Transfer Protocol file system provider	2953
SQL database file system provider	2954
NodeJS file system provider	2955
Google Drive file system provider	2957
Firebase Realtime Database file system provider	2958
IBM Cloud Object Storage file provider	2964
Localization in ##Platform_Name## File manager control	2965
Virtualization in ##Platform_Name## File manager control	2972
Module Injection	2972
Enable Virtualization	2972
Limitations for Virtualization	2974
Accessibility in ##Platform_Name## File Manager component	2974
WAI-ARIA attributes	2975
Keyboard interaction	2976
Ensuring accessibility	2977
See also	2977
Access control in ##Platform_Name## File manager control	2977
Access Rules	2977
Permissions	2978
How To	2981
Adding custom item to context menu in ##Platform_Name## File manager control	2981
Adding custom item to toolbar in ##Platform_Name## File manager control	2982
Enable disable toolbar item in ##Platform_Name## File manager control	2984
Customize custom thumbnail in ##Platform_Name## File manager control	2986
Nested items in ##Platform_Name## File manager control	2987
Change the localization content in ##Platform_Name## File manager control	2992
Create the custom file provider using NodeJS	2993
Floating Action Button	3012
Form validator	3012
Validation rules in ##Platform_Name## Form validator control	3012
Default Rules	3012
Error messages in ##Platform_Name## Form validator control	3013

Customizing Error Messages	3014
Customizing Error Placement.....	3015
Localization in ##Platform_Name## Form validator control.....	3017
Loading translations.....	3017
Gantt	3019
Module in ##Platform_Name## Gantt control.....	3019
Data binding in ##Platform_Name## Gantt control.....	3020
Local data	3020
Remote data.....	3023
Split task.....	3041
Limitations.....	3044
Immutable in ##Platform_Name## Gantt control.....	3044
Limitations.....	3045
Virtual scroll in ##Platform_Name## Gantt control	3046
Row virtualization	3046
Timeline virtualization	3047
Get filtered data when virtual scrolling is enabled.....	3049
Limitations for virtual vcroll	3050
Selection.....	3051
Selection in ##Platform_Name## Gantt control.....	3051
Row selection in ##Platform_Name## Gantt control.....	3057
Cell selection in ##Platform_Name## Gantt control.....	3064
Filtering	3068
Filtering in ##Platform_Name## Gantt control	3068
Filter menu in ##Platform_Name## Gantt control.....	3076
Excel like filter in ##Platform_Name## Gantt control.....	3079
Searching in ##Platform_Name## Gantt control.....	3080
Task dependency in ##Platform_Name## Gantt control	3086
Task relationship types	3086
Define task relationship	3087
Predecessor offset with duration units.....	3088
Disabling automatic dependency offset updates	3090
Validate predecessor links on editing	3092
Dynamically show/hide the dependency line.....	3096
Sorting in ##Platform_Name## Gantt control.....	3098

Sorting column on Gantt initialization	3100
Sorting column dynamically	3101
Clear all the sorted columns dynamically	3103
Sorting events	3104
Sorting Custom Columns.....	3105
Baseline in ##Platform_Name## Gantt control	3107
Timeline.....	3109
Timeline in ##Platform_Name## Gantt control	3109
Top tier and bottom tier in ##Platform_Name## Gantt control	3116
Zooming in ##Platform_Name## Gantt control	3121
Timezone in ##Platform_Name## Gantt control.....	3126
Understanding date manipulation in JavaScript.....	3127
Display same time everywhere with no time difference.....	3127
CRUD operations with timezone.....	3128
Timezone methods	3132
Event markers in ##Platform_Name## Gantt control	3133
Displaying eventMarkers in stacked manner.....	3134
Label positions in ##Platform_Name## gantt control.....	3136
Managing event marker overlapping in ##Platform_Name## gantt control	3138
Work in ##Platform_Name## Gantt control	3139
Task type	3141
Resources in ##Platform_Name## Gantt control.....	3143
Resource collection	3143
Assign resource	3144
Add/Edit resource collection	3146
Resource view in ##Platform_Name## Gantt control	3147
Unassigned task	3147
Resource task	3147
Resource OverAllocation.....	3149
Resource Multi Taskbar	3150
Holidays in ##Platform_Name## Gantt control.....	3156
Tooltip in ##Platform_Name## Gantt control	3157
Enable tooltip.....	3157
Timeline cells tooltip	3160
Cell tooltip.....	3161

Tooltip template	3162
Appearance customization in ##Platform_Name## Gantt control	3168
Taskbar customization	3168
Task labels	3174
Connector lines	3175
Customize rows and cells	3177
Grid lines	3179
Splitter	3180
Duration unit in ##Platform_Name## Gantt control	3184
Duration units	3184
Task scheduling in ##Platform_Name## Gantt control	3188
Automatically Scheduled Tasks	3188
Manually Scheduled Tasks	3189
Custom	3191
Unscheduled Tasks	3193
Define unscheduled tasks in data source	3194
Working Time Range	3196
Weekend/Non-working days	3198
Critical path in ##Platform_Name## Gantt control	3199
Customize taskbar in critical path	3201
Rows in ##Platform_Name## Gantt control	3202
Row height	3202
Expand/Collapse Row	3203
Drag and drop	3207
Customize rows	3216
Styling alternate rows	3217
Row spanning	3218
Scrolling in ##Platform_Name## Gantt control	3219
Set width and height	3219
Responsive with the parent container	3220
Scroll To Date method	3222
Set the vertical scroll position	3223
Managing Tasks	3224
Managing tasks in ##Platform_Name## Gantt control	3224
Adding new tasks in ##Platform_Name## Gantt control	3228

Editing tasks in ##Platform_Name## Gantt control	3232
Validation in ##Platform_Name## Gantt control	3240
Deleting tasks in ##Platform_Name## Gantt control	3244
Task bar editing in ##Platform_Name## Gantt control	3247
In dent and out dent in ##Platform_Name## Gantt control	3259
Splitting and merging tasks in ##Platform_Name## Gantt control	3260
Maintaining data in server in ##Platform_Name## Gantt control	3262
Columns	3268
Columns in ##Platform_Name## Gantt control	3268
Column reordering in ##Platform_Name## Gantt control	3274
Column resizing in ##Platform_Name## Gantt control	3278
Column template in ##Platform_Name## Gantt control	3280
Column menu in ##Platform_Name## Gantt control	3283
Responsive columns in ##Platform_Name## Gantt control	3289
Check box columns in ##Platform_Name## Gantt control	3293
Column spanning in ##Platform_Name## Gantt control	3296
State persistence in ##Platform_Name## Gantt control	3298
Get or set localStorage value	3298
Prevent columns from persisting	3298
Persist the header template and header Text	3300
Tool bar in ##Platform_Name## Gantt control	3302
Built-in toolbar items	3302
Custom toolbar items	3304
Built-in and custom items in toolbar	3305
Enable/disable toolbar items	3306
Add input elements to toolbar	3308
Context menu in ##Platform_Name## Gantt control	3309
Custom context menu items	3310
Excel Export	3312
Excel export in ##Platform_Name## Gantt control	3312
Custom data source in ##Platform_Name## Gantt control	3314
Multiple gantt exporting in ##Platform_Name## Gantt control	3315
Pdf Export	3328
Pdf export in ##Platform_Name## Gantt control	3328
Multiple gantt exporting in ##Platform_Name## Gantt control	3336

To customize PDF export	3337
Customizing header and footer of PDF export in ##Platform_Name## Gantt control	3355
Data markers in ##Platform_Name## Gantt control.....	3362
Global local in ##Platform_Name## Gantt control	3364
Localization	3364
Internationalization.....	3369
Right to left (RTL)	3371
See Also	3374
Accessibility in ##Platform_Name## Gantt control.....	3374
WAI-ARIA attributes.....	3375
Keyboard navigation	3376
Ensuring accessibility	3377
See also	3377
Touch interaction in ##Platform_Name## Gantt control.....	3377
Tooltip	3377
Context menu.....	3377
Sorting.....	3377
Column resize.....	3378
Editing	3379
Selection.....	3381
Loading animation in ##Platform_Name## Gantt control	3382
Style and appearance in ##Platform_Name## Gantt control.....	3384
How To	3385
Open add edit dialog in ##Platform_Name## Gantt control.....	3385
Change schedule date in ##Platform_Name## Gantt control.....	3386
Copy paste records in ##Platform_Name## Gantt control	3388
Maintain record index in ##Platform_Name## Gantt control	3390
Custom field in ##Platform_Name## Gantt control	3393
Maintain zoom to fit in ##Platform_Name## Gantt control	3396
Drag and drop in ##Platform_Name## Gantt control	3399
New row position in ##Platform_Name## Gantt control.....	3401
Render timeline from 1 to 365 days in ##Platform_Name## Gantt control	3403
Grid.....	3406
Getting started in ##Platform_Name## Grid control	3406
Dependencies.....	3406

Setup for local development.....	3406
Configuring system JS	3407
Adding CSS reference.....	3408
Adding Grid component.....	3408
Module injection	3409
Enable paging	3409
Enable sorting	3411
Enable filtering	3413
Enable grouping	3415
Run the application	3417
See Also	3419
Module in ##Platform_Name## Grid control	3419
Data Binding.....	3420
Data binding in ##Platform_Name## Grid control	3420
Local data in ##Platform_Name## Grid control	3423
Remote data in ##Platform_Name## Grid control.....	3427
Immutable mode in ##Platform_Name## Grid control.....	3438
Limitations.....	3442
Performance tips for ##Platform_Name## Grid control	3442
How to improve loading performance by binding large dataset	3443
How to improve loading performance by binding large data by showing custom text or element	3443
How to improve loading performance by referring individual script and CSS	3444
How to update cell values without frequent server calls	3444
How to optimize server-side data operations with adaptors	3444
How to avoid MaxJsonLength error while passing large amount of records	3444
Microsoft excel limitation while exporting millions of records to excel file format.....	3445
Columns	3445
Columns in ##Platform_Name## Grid control.....	3445
Auto generated columns in ##Platform_Name## Grid control.....	3470
Headers in ##Platform_Name## Grid control	3475
Column template in ##Platform_Name## Grid control.....	3510
Complex data binding in ##Platform_Name## Grid control	3517
Foreign key column in ##Platform_Name## Grid control	3519
Auto fit columns in ##Platform_Name## Grid control	3532

Column reorder in ##Platform_Name## Grid control	3534
Column resizing in ##Platform_Name## Grid control	3543
Column chooser in ##Platform_Name## Grid control	3553
Column menu in ##Platform_Name## Grid control	3562
Column spanning in ##Platform_Name## Grid control	3573
Responsive columns in ##Platform_Name## Grid control	3581
Row	3582
Row in ##Platform_Name## Grid control	3582
Row height in ##Platform_Name## Grid control	3591
Row template in ##Platform_Name## Grid control	3594
Detail template in ##Platform_Name## Grid control	3601
Row drag and drop in ##Platform_Name## Grid control	3610
Row spanning in ##Platform_Name## Grid control	3620
Cell	3624
Cell in ##Platform_Name## Grid control	3624
Content in ##Platform_Name## Grid control	3630
Auto wrap in ##Platform_Name## Grid control	3631
Clip mode in ##Platform_Name## Grid control	3633
Editing	3634
Edit in ##Platform_Name## Grid control	3634
Edit types in ##Platform_Name## Grid control	3647
In line editing in ##Platform_Name## Grid control	3672
Dialog editing in ##Platform_Name## Grid control	3689
Template editing in ##Platform_Name## Grid control	3701
Batch editing in ##Platform_Name## Grid control	3711
Command column editing in ##Platform_Name## Grid control	3723
Validation in ##Platform_Name## Grid control	3727
Persisting data in server in ##Platform_Name## Grid control	3734
Sorting in ##Platform_Name## Grid control	3740
Initial sort	3742
Multi-column sorting	3744
Sort order	3746
Sort foreign key column based on Text	3746
Sorting events	3748
Custom sort comparer	3750

Touch interaction	3751
Grouping	3752
Grouping in ##Platform_Name## Grid control.....	3752
Caption template in ##Platform_Name## Grid control	3764
Reordering on grouped columns in ##Platform_Name## Grid control	3769
Lazy load grouping in ##Platform_Name## Grid control	3771
Filtering	3776
Filtering in ##Platform_Name## Grid control	3776
Filter bar in ##Platform_Name## Grid control	3783
Filter menu in ##Platform_Name## Grid control	3789
Excel like filter in ##Platform_Name## Grid control	3800
Searching in ##Platform_Name## Grid control.....	3805
Initial search	3806
Search operators.....	3808
Search by external button.....	3808
Search specific columns	3810
Clear search by external button.....	3811
Search on each key stroke	3813
Highlight the search text.....	3814
Perform search operation in Grid using multiple keywords.....	3816
See Also	3819
Paging in ##Platform_Name## Grid control.....	3819
Template	3821
Pager with Page Size Dropdown	3823
How to render Pager at the Top of the Grid.....	3825
See Also	3827
Scrolling in ##Platform_Name## Grid control.....	3827
Set width and height.....	3827
Responsive with parent container	3829
Sticky Header	3831
Scroll to selected row.....	3832
Hide the scrollbar when the content is not overflown.....	3834
Frozen in ##Platform_Name## Grid control.....	3835
Limitations of Frozen Grid.....	3837
Freeze particular columns.....	3837

Freeze Direction	3839
Deprecated Methods	3840
Virtual scroll in ##Platform_Name## Grid control	3842
Row Virtualization	3842
Column Virtualization	3844
Virtualization with Grouping	3848
Limitations for virtual scrolling	3848
Browser height limitation in virtual scrolling and solution	3849
Infinite scroll in ##Platform_Name## Grid control	3855
InitialBlocks	3858
Cache Mode	3860
Limitations for Infinite Scrolling	3862
Selection	3863
Selection in ##Platform_Name## Grid control	3863
Row selection in ##Platform_Name## Grid control	3867
Cell selection in ##Platform_Name## Grid control	3877
Column selection in ##Platform_Name## Grid control	3880
Check box selection in ##Platform_Name## Grid control	3882
Aggregates	3888
Aggregates in ##Platform_Name## Grid control	3888
Footer aggregate in ##Platform_Name## Grid control	3888
Group and caption aggregate in ##Platform_Name## Grid control	3894
Custom aggregate in ##Platform_Name## Grid control	3896
Reactive aggregate in ##Platform_Name## Grid control	3900
Print in ##Platform_Name## Grid control	3904
Page setup	3906
Print using an external button	3906
Print the visible page	3907
Print the hierarchy grid	3909
Print the master detail grid	3911
Print large number of columns	3913
Show or Hide columns while Printing	3913
Limitations of Printing Large Data	3915
See Also	3915
Adaptive in ##Platform_Name## Grid control	3915

Render adaptive dialogs.....	3916
Vertical row rendering	3917
Hierarchy grid in ##Platform_Name## Grid control.....	3921
ExpandAll by external button	3923
Expand child grid initially	3925
Dynamically load child grid data	3927
Bind hierarchy grid with different field.....	3929
Adding Record in ChildGrid	3931
Dynamically bind data to child grid based on parent row Data	3933
State Persistence.....	3935
State persistence in ##Platform_Name## Grid control.....	3935
Get or set local storage value in ##Platform_Name## Grid control.....	3940
Prevent to persist in ##Platform_Name## Grid control	3940
Add to persist in ##Platform_Name## Grid control	3942
Tool Bar	3944
Tool bar in ##Platform_Name## Grid control	3944
Tool bar items in ##Platform_Name## Grid control	3948
Custom tool bar in ##Platform_Name## Grid control.....	3958
Pdf Export.....	3962
Pdf export in ##Platform_Name## Grid control.....	3962
Export multiple grids in ##Platform_Name## Grid control	3971
Pdf export options in ##Platform_Name## Grid control.....	3975
Pdf cell style customization in ##Platform_Name## Grid control.....	3996
Adding header and footer in ##Platform_Name## Grid control	4001
Exporting hierarchy grid in ##Platform_Name## Grid control.....	4009
Exporting grid with templates in ##Platform_Name## Grid control.....	4011
Exporting grid in server in ##Platform_Name## Grid control	4022
Excel Export.....	4029
Excel exporting in ##Platform_Name## Grid control	4029
Export multiple grids in ##Platform_Name## Grid control	4039
Excel export options in ##Platform_Name## Grid control.....	4043
Excel cell style customization in ##Platform_Name## Grid control.....	4061
Adding header and footer in ##Platform_Name## Grid control	4068
Exporting hierarchy grid in ##Platform_Name## Grid control.....	4071
Exporting grid with templates in ##Platform_Name## Grid control.....	4073

Exporting grid in server in ##Platform_Name## Grid control	4083
Global local in ##Platform_Name## Grid control.....	4093
Localization	4093
Internationalization.....	4098
Right to left (RTL)	4101
See Also	4103
Accessibility in ##Platform_Name## Grid control	4103
WAI-ARIA attributes.....	4103
Keyboard interaction	4104
Ensuring accessibility	4105
See also	4105

Browser support

The Syncfusion Essential JS 2 components are supported only in modern browsers. This includes the following versions.

Chrome	Firefox	Opera	Edge	IE	Safari	iOS	Android	Windows Mobile
63+	58+	50+	13+	11+	9+	9+	4.4+	IE 11+

Required polyfills

The following polyfills are required to run Essential JS 2 components in each browser.

Browser	Polyfills
Chrome(latest), Firefox(latest), Opera(latest), Edge, Safari 9+	NONE
IE 11	ES6 Promise

The Syncfusion Essential JS 2 components are supported in IE 11 browser with ES6 Promise polyfill.

Using CDN

To add ES6 Promise polyfill using CDN, include this in your HTML file.

```
`ts
<!-- Automatically provides/replaces Promise if missing or broken. -->
<script src="https://cdn.jsdelivr.net/npm/es6-promise@4/dist/es6-promise.js"></script>
<script src="https://cdn.jsdelivr.net/npm/es6-promise@4/dist/es6-promise.auto.js"></script>
<!-- Minified version of es6-promise-auto below. -->
<script src="https://cdn.jsdelivr.net/npm/es6-promise@4/dist/es6-promise.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/es6-promise@4/dist/es6-promise.auto.min.js"></script>
`
```

Node.js

ES6 Promise polyfill can also be installed in node.js.

To install:

```
`ts
yarn add es6-promise
(or)
npm install es6-promise
`
```

To Use:

```
`ts  
var Promise = require('es6-promise').Promise;  
`
```

For further details, refer to the link [here](#).

Getting started

Installation and upgrade

<!-- markdownlint-disable MD024 -->

Installation

Install by using npm CLI

Syncfusion JavaScript (Essential JS 2) packages are published in [npm](#). You can install the necessary packages from npm's install command. For example, grid package can be installed using following command.

```
`  
npm install @syncfusion/ej2-grids --save  
`
```

Install by using package.json

1. Add the Syncfusion JavaScript (Essential JS 2) package references in the **dependencies** of **~/package.json** file.

```
`  
{  
  "dependencies": {  
    "@syncfusion/ej2-grids": "*",  
    "@syncfusion/ej2-charts": "*"  
  }  
}  
`
```

The ***** indicates the latest version of npm package. Refer the [documentation](#) for more details about npm versioning.

2. Now, open the command prompt and run the **npm install** command line. This will install all npm dependencies in a single command line.

Refer the [documentation](#) for more details about npm package.json

Download JavaScript – EJ2 Installer

The Syncfusion JavaScript - EJ2 installer can be downloaded from the Syncfusion website. You can either download the licensed installer or try our trial installer depending on your license.

- Trial Installer
- Licensed Installer

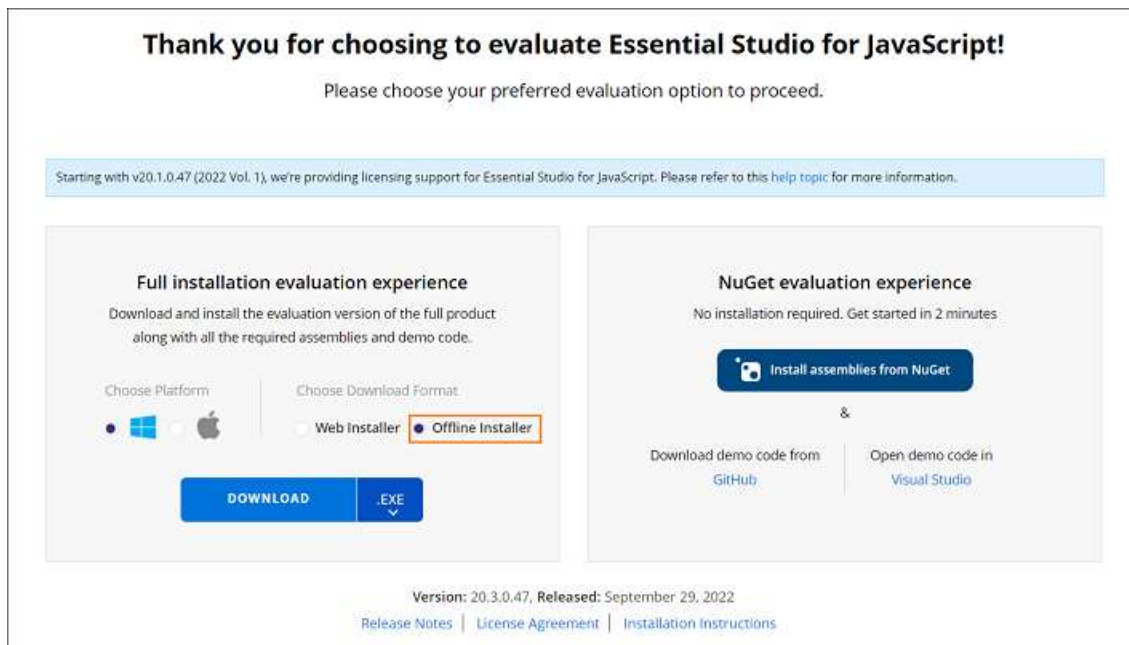
Download the Trial Version

Our 30-day trial can be downloaded in two ways.

- Download Free Trial Setup
- Start Trials if using components through [npm](#)

Download Free Trial Setup

1. You can evaluate our 30-day free trial by visiting the [Download Free Trial](#) page and select the JavaScript platform.
2. After completing the required form or logging in with your registered Syncfusion account, you can download the JavaScript - EJ2 trial installer from the confirmation page. (See the screenshot below.)



3. With a trial license, only the latest version's trial installer can be downloaded.
4. After downloading, the Syncfusion JavaScript - EJ2 trial installer can be unlocked using either the trial unlock key or the Syncfusion registered login credential. More information on generating an unlock key can be found in this article.
5. Before the trial expires, you can download the trial installer at any time from your registered account's Trials & Downloads page (See the screenshot below.)
6. Click the Download (element 1 in the screenshot below) button to get the Syncfusion Essential Studio JavaScript – EJ2 web installer.

Trial Downloads and Unlock Keys

Starting with v20.1.0.47 (2022 Vol. 1), we're providing licensing support for Essential Studio for JavaScript(Essential JS 2). Please refer to this [help topic](#) for more information.

JavaScript (Essential JS 2) [Redacted]

Trial Version : [Redacted]

[Release Notes](#) | [Get License Key](#) ⓘ | [Get Unlock Key](#) ⓘ | [Security Management Report](#)

[Download](#) 1

[More Download Options](#) 2

- Click the More Download Options (element 2 in the above screenshot) button to get the Essential Studio JavaScript – EJ2 Offline trial installer which is available in EXE and ZIP format.

Downloads

Version : 20.3.0.47
Released on : Sep 29, 2022

[Windows](#) [Mac](#)

WEB - Essential JS 2 Offline Installer Web Installer

JavaScript - EJ 2
Includes Angular, React, and Vue components.
[Release Notes](#) | [Readme](#)

[Download](#) .EXE 488 MB Checksum Value [Download](#) .ZIP 488 MB Checksum Value [Download](#) .EXE 3 MB Checksum Value

Start Trials if using components through [npm](#)

You should initiate an evaluation if you have already obtained our components through [npm](#)

- You can start your 30-day free trial for JavaScript – EJ2 from the [Start Trial](#) page from your account.

WEB

Blazor
Includes 80+ Blazor components for ASP.NET Core application development.
[Trial in Progress](#)

JavaScript
Includes 80+ JavaScript controls for developing modern web applications. It also includes complete support for Angular, React, and Vue frameworks.
[Start Trial](#)

- To access this page, you must sign up\log in with your Syncfusion account.
- Begin your trial by selecting the JavaScript – EJ2 product.

Note: If you've already used the trial products and they haven't expired, you won't be able to start the trial for the same product again.

- After you've started the trial, go to the [Trials & Downloads](#) page to get the latest version trial installer. You can generate the [unlock](#) key here at any time before the trial period expires. (See the screenshot below.)

Trial Downloads and Unlock Keys



- You can find your current active trial products on the [Trials & Downloads](#) page.

Download the License Version

- Syncfusion licensed products will be available in the [License & Downloads](#) page under your registered Syncfusion account.
- You can view all the licenses (both active and expired) associated with your account.
- Click the Download (element 1 in the screenshot below) button to download the respective product's installer.
- The most recent version of the installer will be downloaded from this page.
- To download older version installers, go to [Downloads Older Versions](#) (element 2 in the screenshot below).
- You can download other platform\add-on installers by going to More Downloads Options (element 3 in the screenshot below).
- For Windows OS, EXE and Zip formats are available for download. They are both Offline Installers.



You can also refer to the [Online installer](#) and [Offline installer](#) links for step-by-step installation guidelines.

Installation using Web Installer

You can refer to the [Download](#) section to learn how to get the JavaScript – EJ2 trial or licensed installer.

Overview

For the Essential Studio JavaScript – EJ2 product, Syncfusion offers a Web Installer. This installer alleviates the burden of downloading a larger installer. You can simply download and run the online installer, which will be smaller in size and will download and install the Essential Studio products you have chosen. You can get the most recent version of Essential Studio Web Installer [here](#).

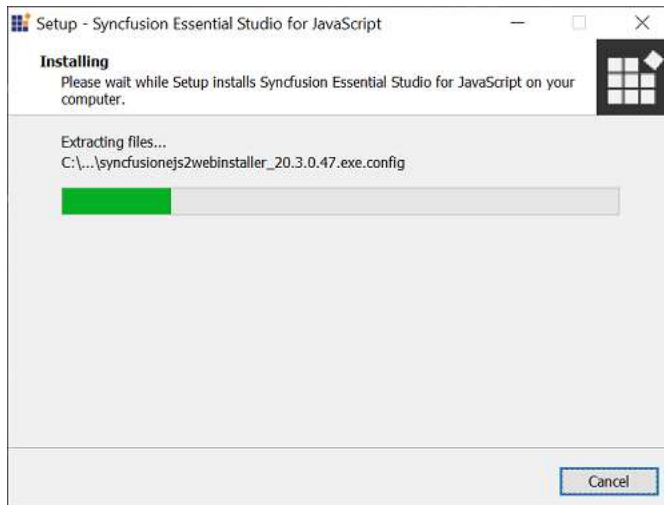
The frameworks listed below are supported in this installer.

- JavaScript
- Angular
- React
- Vue
- JavaScript (ES5)

Installation

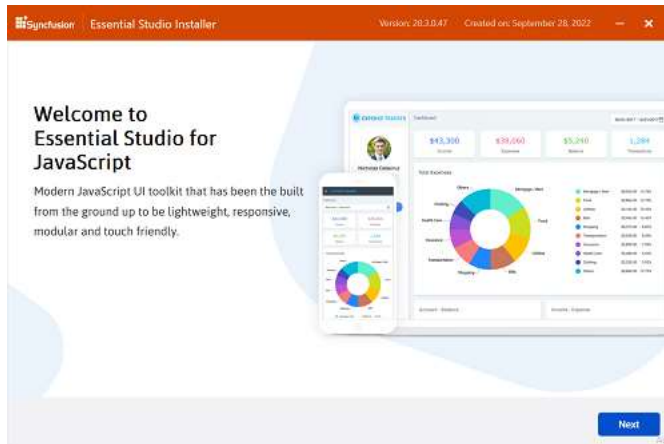
The steps below show how to install Essential Studio JavaScript – EJ2 Web Installer.

1. Open the Syncfusion Essential Studio JavaScript – EJ2 Web Installer file from downloaded location by double-clicking it. The Installer Wizard automatically opens and extracts the package.



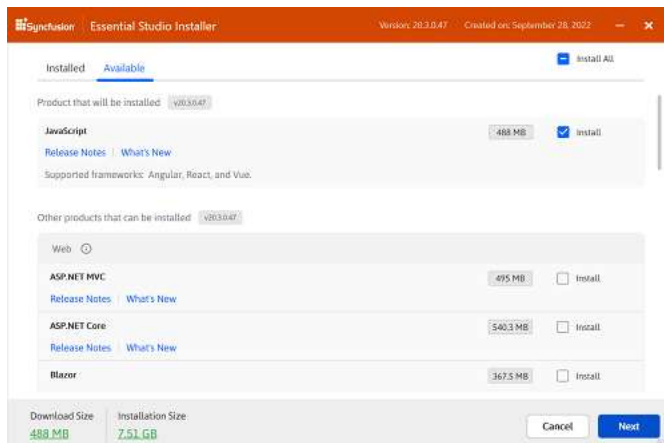
Note: The installer wizard extracts the syncfusionejs2webinstaller_{version}.exe dialog, which displays the package's unzip operation.

2. The Syncfusion JavaScript - EJ2 Web Installer's welcome wizard will be displayed. Click the Next button.



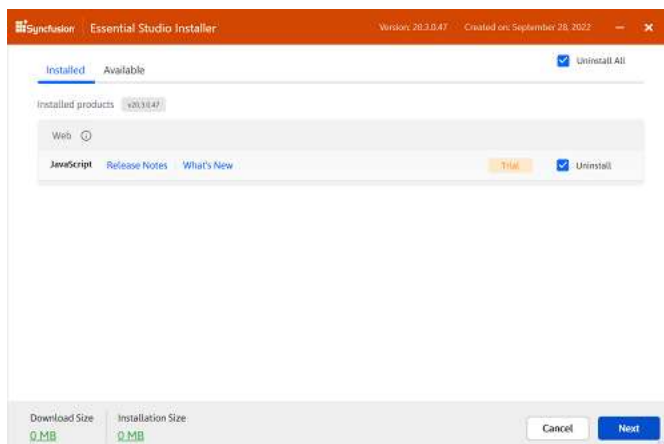
3. The Platform Selection Wizard will appear. From the **Available** tab, select the products to be installed. Select the Install All checkbox to **install all** products.

Available



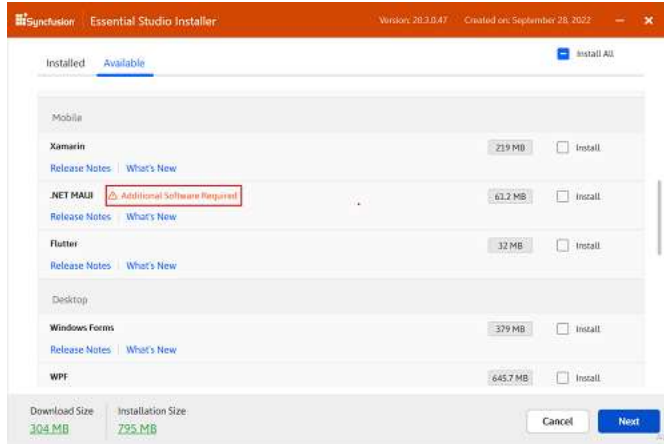
If you have multiple products installed in the same version, they will be listed under the **Installed** tab. You can also select which products to uninstall from the same version. Click the Next button.

Installed

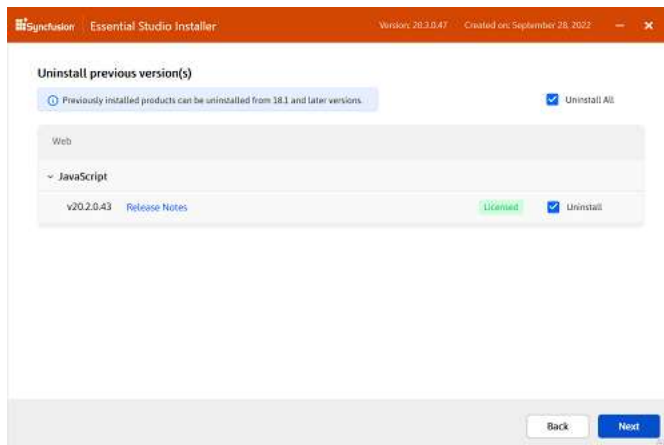


Important: If the required software for the selected product isn't already installed, the **Additional Software Required** alert will appear. You can, however, continue the installation and install the necessary software later.

Required Software

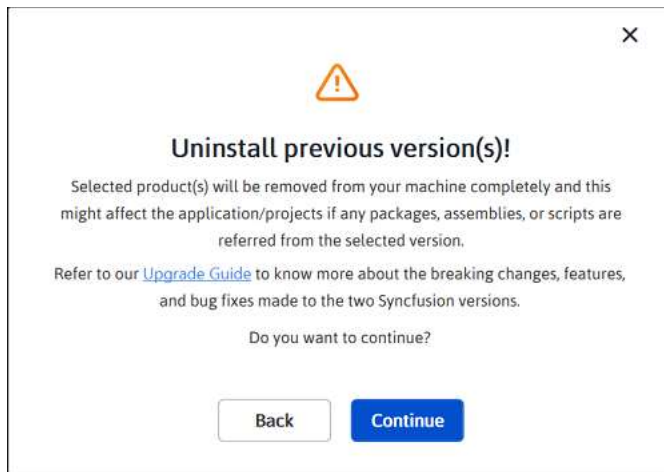


- If previous version(s) for the selected products are installed, the Uninstall previous version wizard will be displayed. You can see the list of previously installed versions for the products you've chosen here. To remove all versions, check the **Uninstall All** checkbox. Click the Next button.

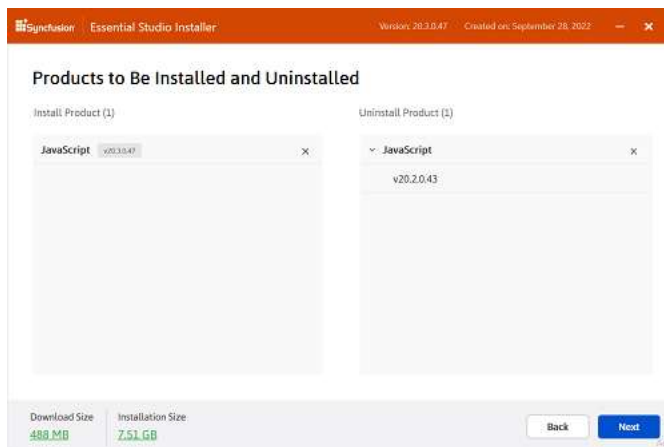


Note: From the 2021 Volume 1 release, Syncfusion has provided option to uninstall the previous versions from 18.1 while installing the new version.

- Pop up screen will be displayed to get the confirmation to uninstall selected previous versions.

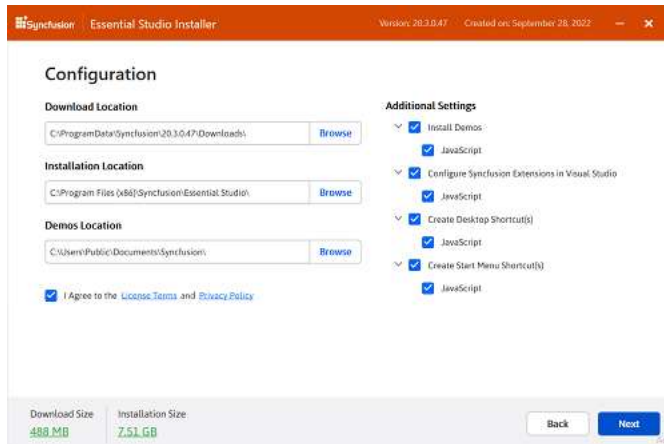


6. The Confirmation Wizard will appear with the list of products to be installed/uninstalled. You can view and modify the list of products that will be installed and uninstalled from this page.



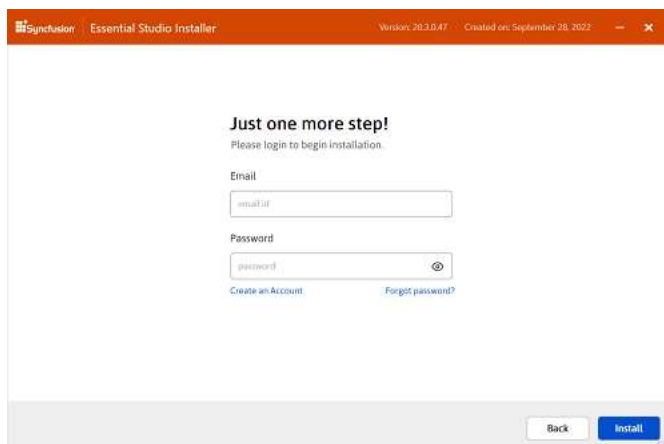
Note: By clicking the **Download Size** and **Installation Size** links, you can determine the approximate size of the download and installation

7. The Configuration Wizard will appear. You can change the Download, Install, and Demos locations from here. You can also change the Additional settings on a product-by-product basis. Click Next to install with the default settings.



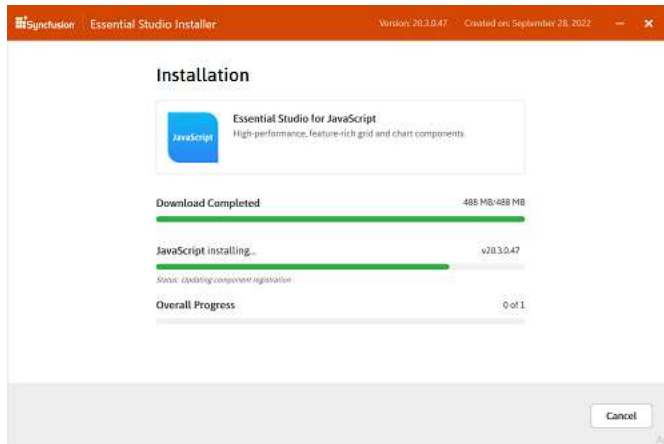
Additional settings

- Select the **Install Demos** check box to install Syncfusion samples, or leave the check box unchecked, if you do not want to install Syncfusion samples
 - Select the **Configure Syncfusion Extensions controls in Visual Studio** checkbox to configure the Syncfusion Extensions in Visual Studio or clear this check box when you do not want to configure the Syncfusion Extensions in Visual Studio.
 - Check the **Create Desktop Shortcut** checkbox to add a desktop shortcut for Syncfusion Control Panel
 - Check the **Create Start Menu Shortcut** checkbox to add a shortcut to the start menu for Syncfusion Control Panel
8. After reading the License Terms and Conditions, check the **I agree to the License Terms and Privacy Policy** check box. Click the Next button.
 9. The login wizard will appear. You must enter your Syncfusion email address and password. If you do not already have a Syncfusion account, you can create one by clicking on **Create an Account**. If you have forgotten your password, click **Forgot Password** to create a new one. Click the Install button.

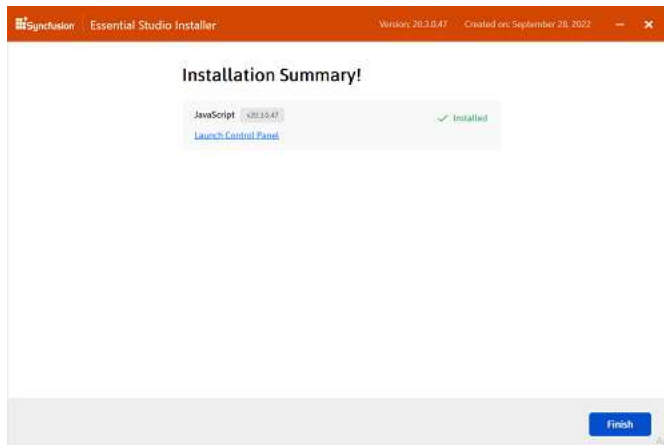


Important: The products you have chosen will be installed based on your Syncfusion License (Trial or Licensed).

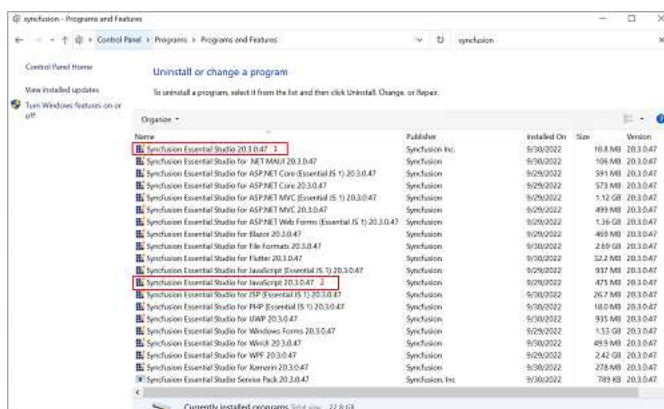
10. The download and installation\uninstallation progress will be displayed as shown below.



11. When the installation is finished, the **Summary** wizard will appear. Here you can see the list of products that have been installed successfully and those that have failed. To close the Summary wizard, click Finish.



- To open the Syncfusion Control Panel, click **Launch Control Panel**.
12. After installation, there will be two Syncfusion control panel entries, as shown below. The Essential Studio entry will manage all Syncfusion products installed in the same version, while the Product entry will only uninstall the specific product setup.



Uninstallation

Syncfusion JavaScript – EJ2 installer can be uninstalled in two ways.

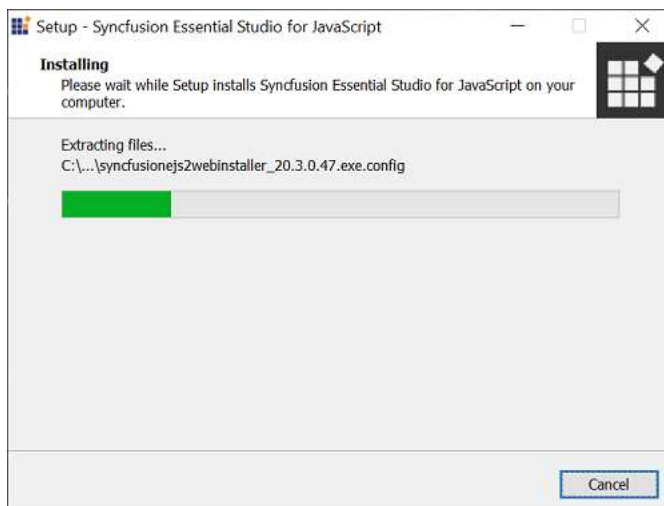
- Uninstall the JavaScript – EJ2 using the Syncfusion JavaScript – EJ2 web installer
- Uninstall the JavaScript – EJ2 from Windows Control Panel

Follow either one of the option below to uninstall Syncfusion Essential Studio JavaScript – EJ2 installer

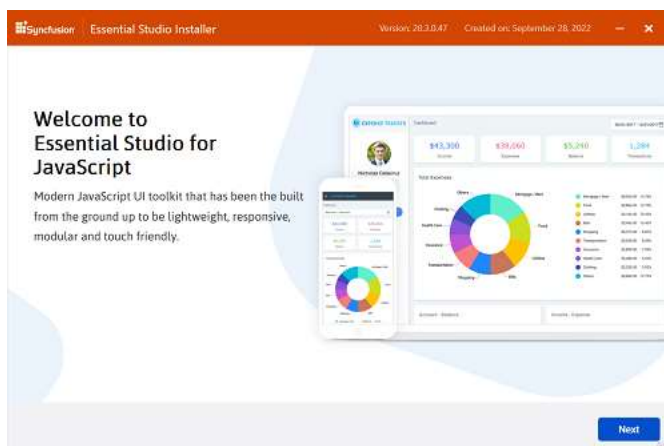
Option 1: Uninstall the JavaScript–EJ2 using the Syncfusion JavaScript–EJ2 web installer

Syncfusion provides the option to uninstall products of the same version directly from the Web Installer application. Select the products to be uninstalled from the list, and Web Installer will uninstall them one by one.

Open the Syncfusion Essential Studio JavaScript – EJ2 Online Installer file from downloaded location by double-clicking it. The Installer Wizard automatically opens and extracts the package



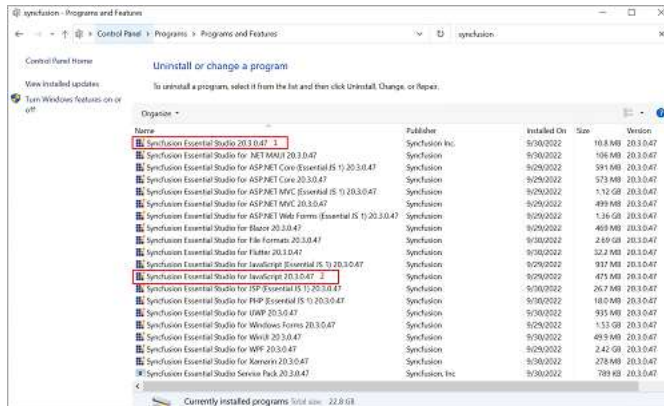
The Syncfusion JavaScript – EJ2 Web Installer’s welcome wizard will be displayed. Click the Next button



Option 2: Uninstall the JavaScript–EJ2 from Windows Control Panel

You can uninstall all the installed products by selecting the **Syncfusion Essential Studio {version}** entry (element 1 in the below screenshot) from the Windows control panel, or you can uninstall JavaScript –

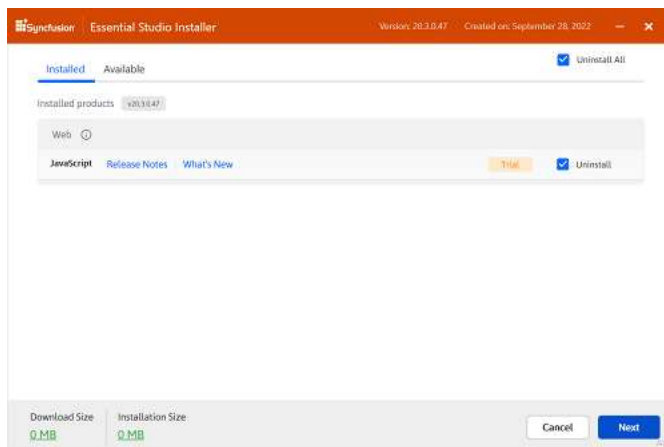
EJ2 alone by selecting the **Syncfusion Essential Studio for JavaScript – EJ2 {version}** entry (element 2 in the below screenshot) from the Windows control panel.



Note: If the **Syncfusion Essential Studio for JavaScript {version}** entry is selected from the Windows control panel, the Syncfusion Essential Studio JavaScript – EJ2 alone will be removed and the below default MSI uninstallation window will be displayed.

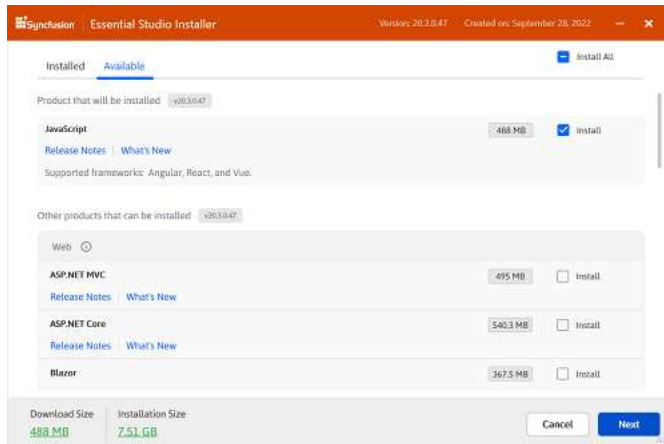
1. The Platform Selection Wizard will appear. From the **Installed** tab, select the products to be uninstalled. To select all products, check the **Uninstall All** checkbox. Click the Next button.

Installed

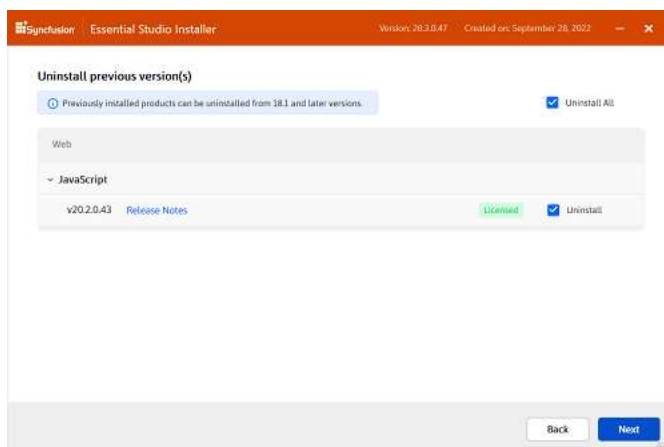


You can also select the products to be installed from the **Available** tab. Click the Next button.

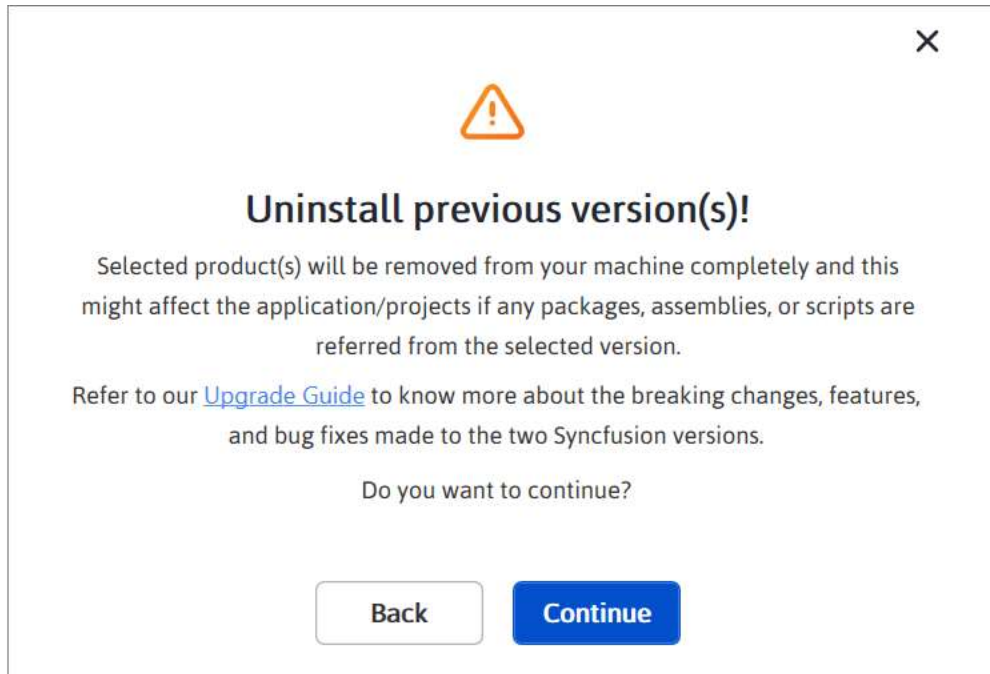
Available



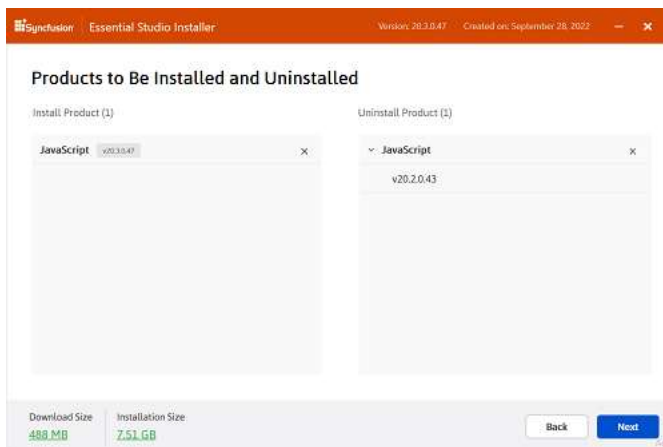
2. If any other products selected for installation, Uninstall previous version wizard will be displayed with previous version(s) installed for the selected products. Here you can view the list of installed previous versions for the selected products. Select **Uninstall All** checkbox to select all the versions. Click Next.



3. Pop up screen will be displayed to get the confirmation to uninstall selected previous versions.

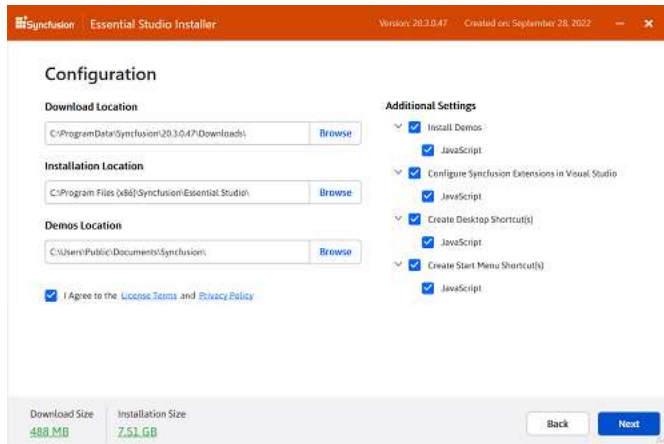


4. The Confirmation Wizard will appear with the list of products to be installed/uninstalled. Here you can view and modify the list of products that will be installed/uninstalled.

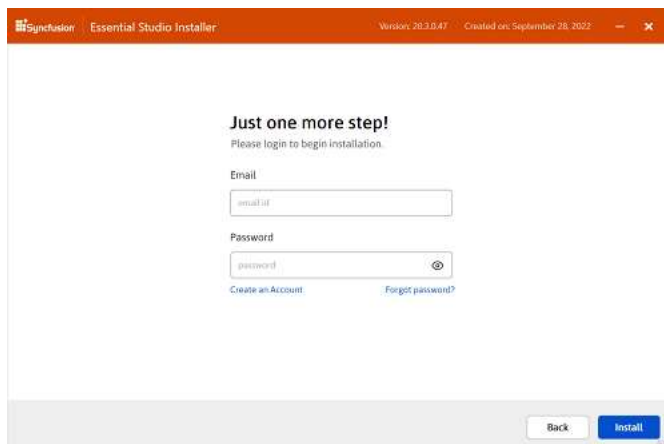


Note: By clicking the **Download Size** and **Installation Size** links, you can determine the approximate size of the download and installation

5. The Configuration Wizard will appear. You can change the Download, Install, and Demos locations from here. You can also change the Additional settings on a product-by-product basis. Click Next to install with the default settings.

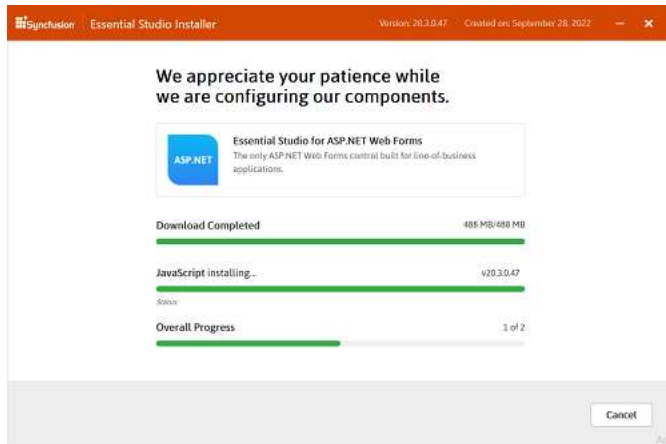


6. After reading the License Terms and Conditions, check the **I agree to the License Terms and Privacy Policy** check box. Click the Next button.
7. The login wizard will appear. You must enter your Syncfusion email address and password. If you do not already have a Syncfusion account, you can create one by clicking on **Create an Account**. If you have forgotten your password, click **Forgot Password** to create a new one. Click the Install button.

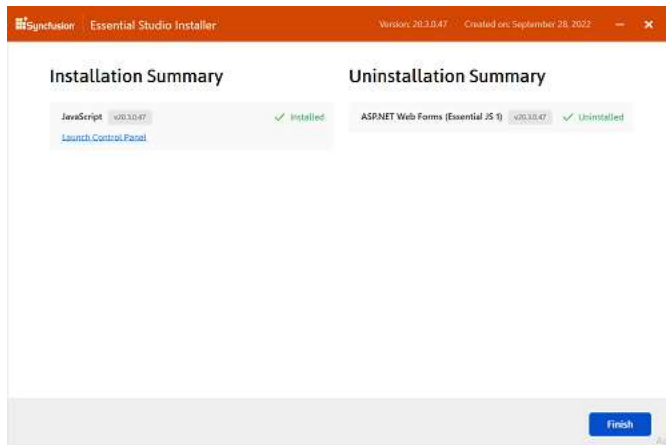


Important: The products you have chosen will be installed based on your Syncfusion License (Trial or Licensed).

8. The download, installation, and uninstallation progresses will be shown.



9. When the installation is finished, the **Summary** wizard will appear. Here you can see the list of products that have been successfully and unsuccessfully installed/uninstalled. To close the Summary wizard, click Finish.



- To open the Syncfusion Control Panel, click **Launch Control Panel**

Installation using Offline Installer

You can refer to the [Download](#) section to learn how to get the JavaScript – EJ2 trial or licensed installer.

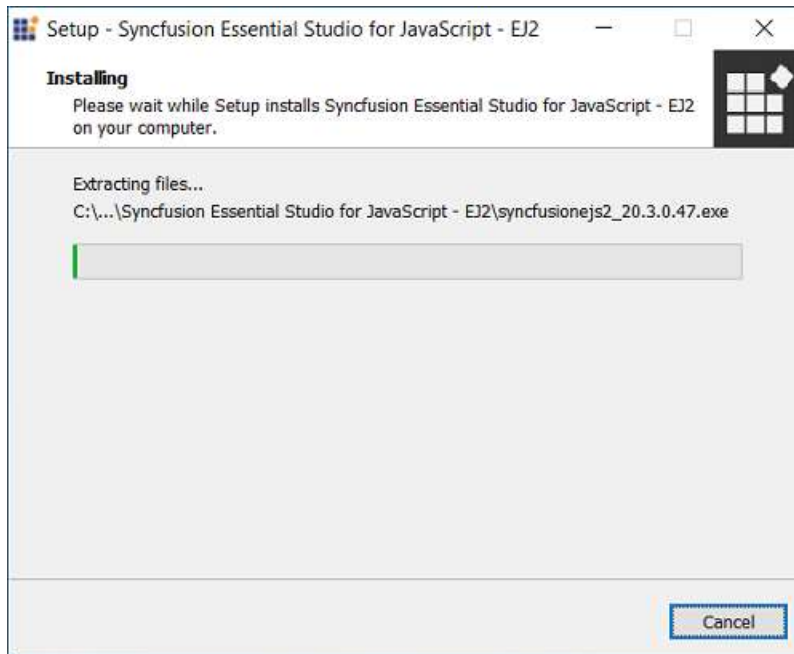
The frameworks listed below are supported in this installer.

- JavaScript
- Angular
- React
- Vue
- JavaScript (ES5)

Installing with UI

The steps below show how to install the Essential Studio JavaScript – EJ2 installer.

1. Open the Syncfusion JavaScript – EJ2 offline installer file from downloaded location by double-clicking it. The Installer Wizard automatically opens and extracts the package

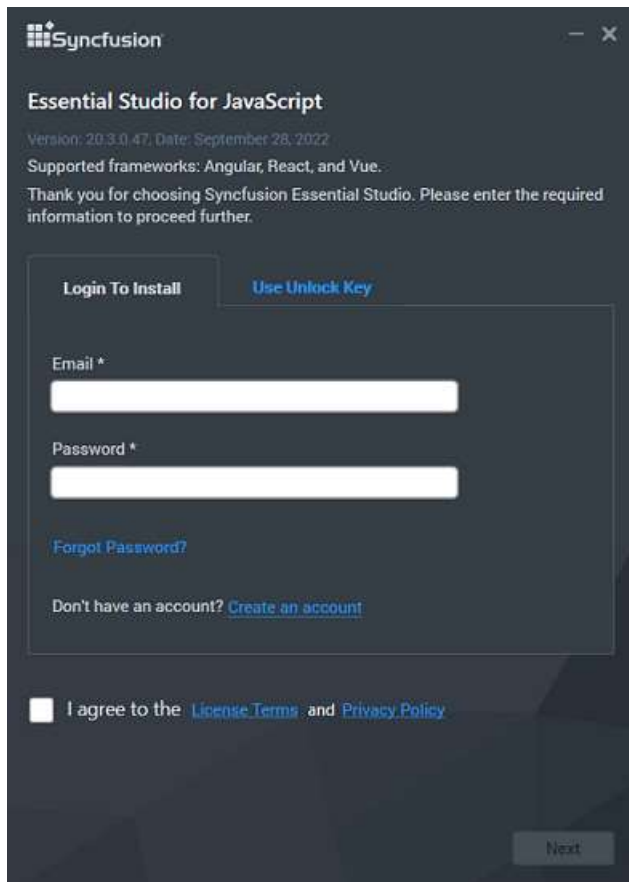


Note: The Installer wizard extracts the syncfusionejs2 (version).exe dialog, which displays the package's unzip operation.

2. To unlock the Syncfusion offline installer, you have two options:
 - Login To Install
 - Use Unlock Key

Login To Install

You must enter your Syncfusion email address and password. If you don't already have a Syncfusion account, you can sign up for one by clicking **"Create an account"**. If you have forgotten your password, click on **"Forgot Password"** to create a new one. Once you've entered your Syncfusion email and password, click Next.

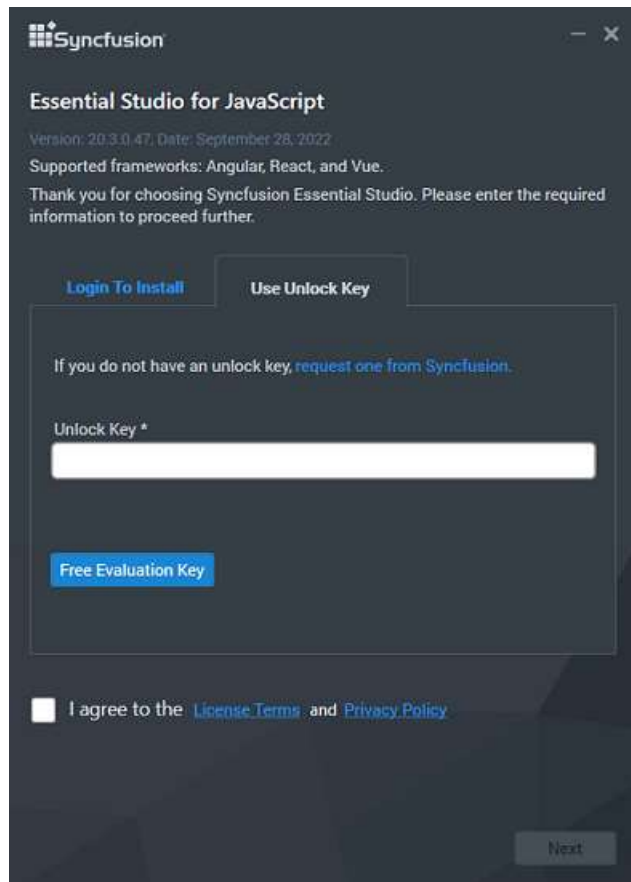


Use Unlock Key

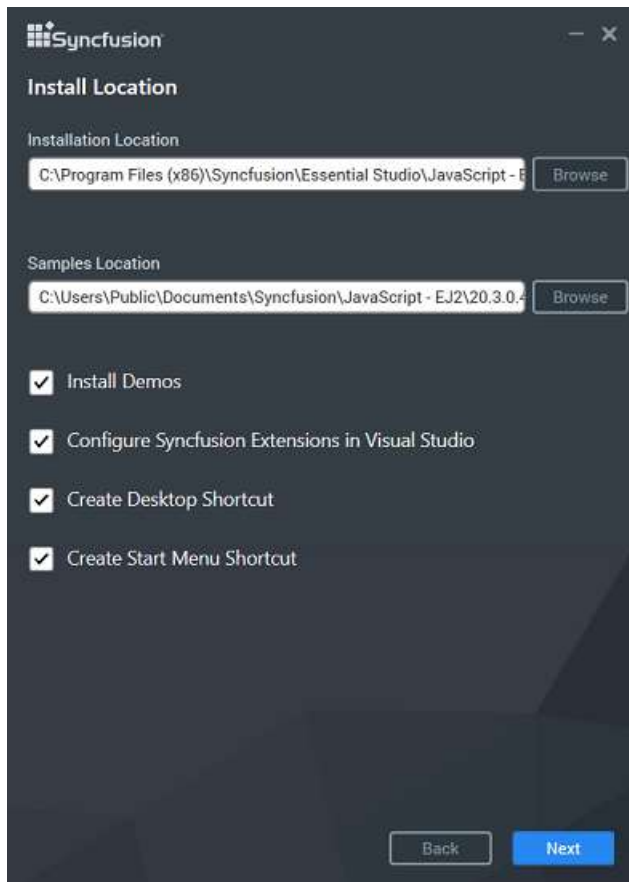
Unlock keys are used to unlock the Syncfusion offline installer, and they are platform and version specific. You should use either Syncfusion licensed or trial Unlock key to unlock Syncfusion JavaScript – EJ2 installer.

The trial unlock key is only valid for 30 days, and the installer will not accept an expired trial key.

To learn how to generate an unlock key for both trial and licensed products, see [this](#) Knowledge Base article.

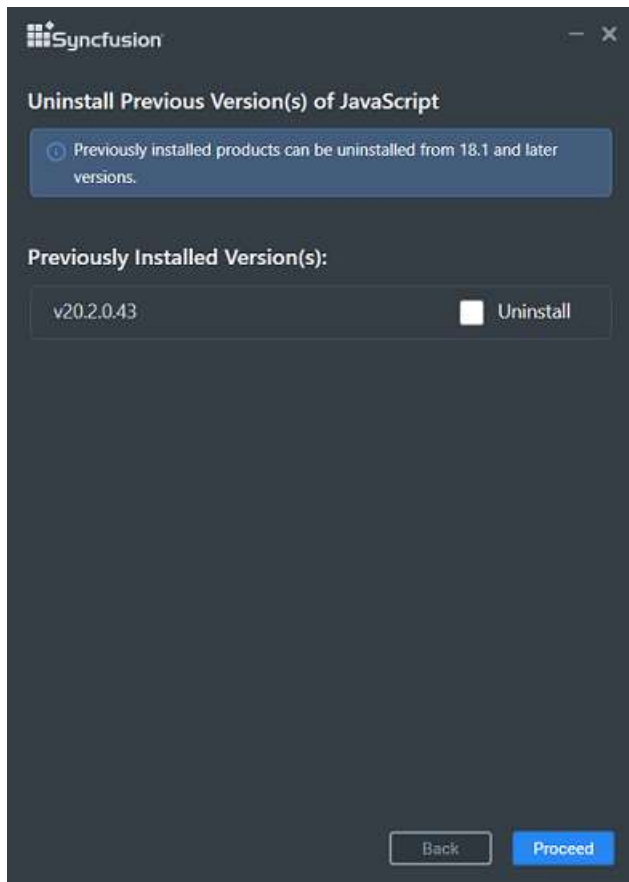


3. After reading the License Terms and Privacy Policy, check the **“I agree to the License Terms and Privacy Policy”** check box. Click the Next button.
4. Change the install and sample locations here. You can also change the Additional settings. Click Next\Install to install with the default settings.



Additional Settings

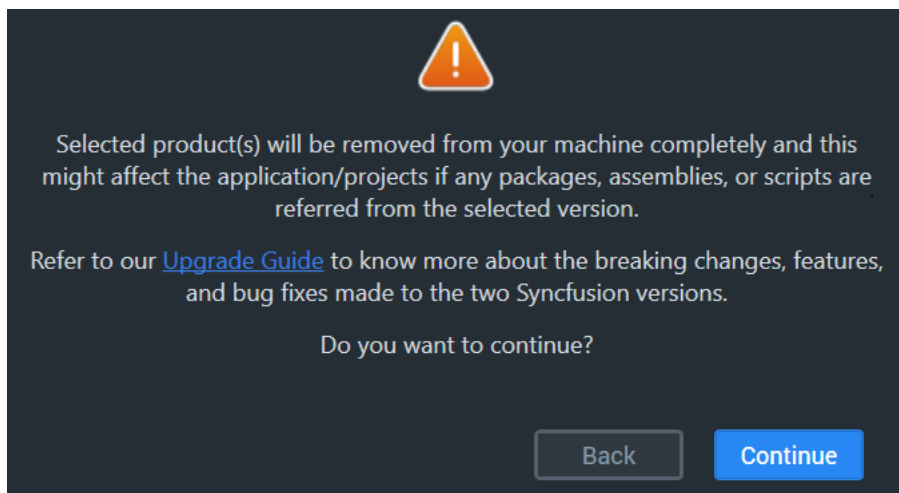
- Select the **Install Demos** check box to install Syncfusion samples, or leave the check box unchecked, if you do not want to install Syncfusion samples
- Select the **Configure Syncfusion Extensions controls in Visual Studio** checkbox to configure the Syncfusion Extensions in Visual Studio or clear this check box when you do not want to configure the Syncfusion Extensions in Visual Studio.
- Check the **Create Desktop Shortcut** checkbox to add a desktop shortcut for Syncfusion Control Panel
- Check the **Create Start Menu Shortcut** checkbox to add a shortcut to the start menu for Syncfusion Control Panel
- 5. If any previous versions of the current product is installed, the **Uninstall Previous Version(s)** wizard will be opened. Select Uninstall checkbox to uninstall the previous versions and then click the Proceed button.



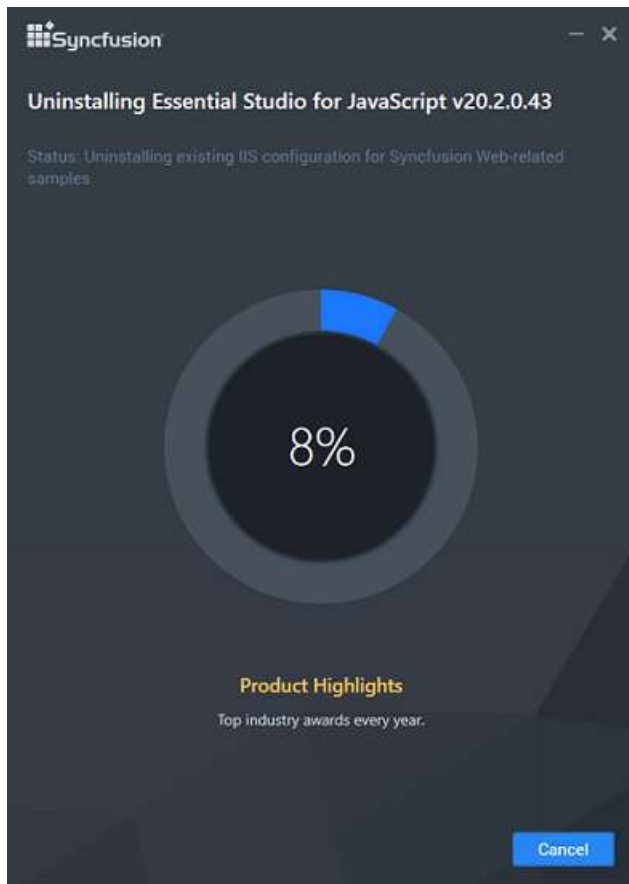
Note: From the 2021 Volume 1 release, Syncfusion has added the option to uninstall previous versions from 18.1 while installing the new version.

Note: If any version is selected to uninstall, a confirmation screen will appear; if continue is selected, the Progress screen will display the uninstall and install progress, respectively. If none of the versions are chosen to be uninstalled, only the installation progress will be displayed.

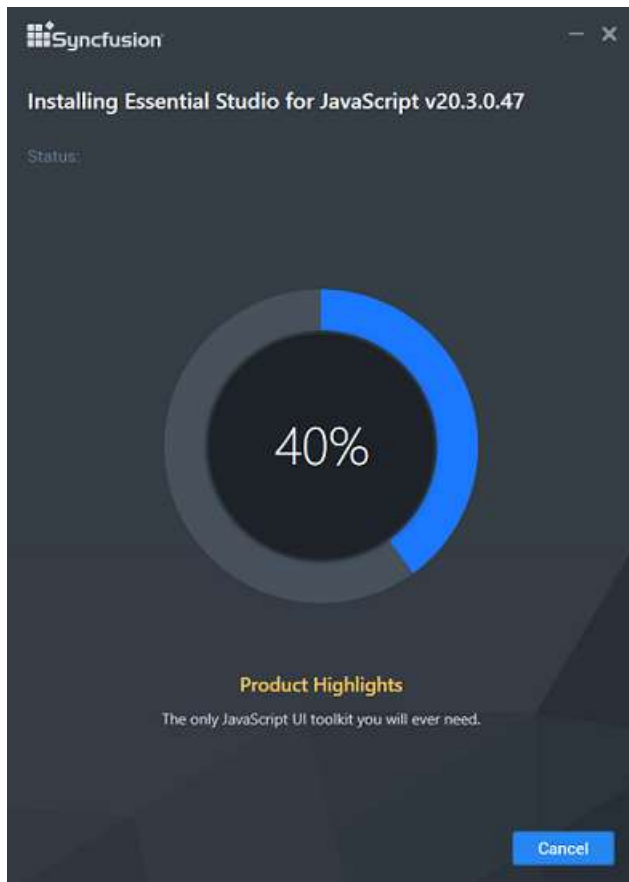
Confirmation Alert



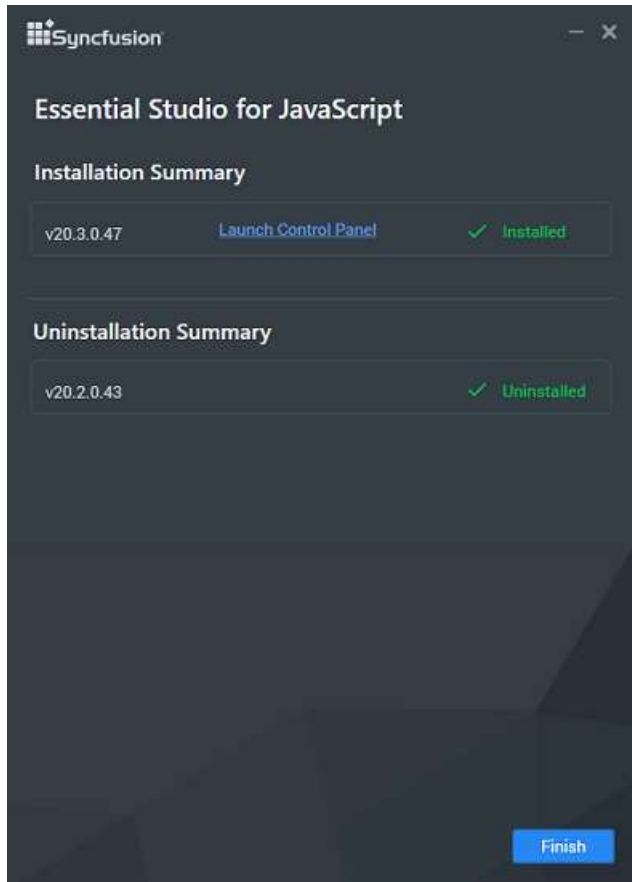
Uninstall Progress:



Install Progress



Note: The Completed screen is displayed once the JavaScript – EJ2 product is installed. If any version is selected to uninstall, The completed screen will display both install and uninstall status.



6. After installing, click the Launch Control Panel link to open the Syncfusion Control Panel.
7. Click the Finish button. Your system has been installed with the Syncfusion Essential Studio JavaScript – EJ2 product.

Installing in silent mode

The Syncfusion Essential Studio JavaScript – EJ2 Installer supports installation and uninstallation via the command line.

Command Line Installation

To install through the Command Line in Silent mode, follow the steps below.

1. Run the Syncfusion JavaScript – EJ2 installer by double-clicking it. The Installer Wizard automatically opens and extracts the package.
2. The file syncfusionejs2_(version).exe file will be extracted into the Temp directory.
3. Run %temp%. The Temp folder will be opened. The syncfusionejs2_(version).exe file will be located in one of the folders.
4. Copy the extracted syncfusionejs2_(version).exe file in local drive.
5. Exit the Wizard.
6. Run Command Prompt in administrator mode and enter the following arguments.

Arguments: "installer file path\SyncfusionEssentialStudio(product)_(version).exe" /Install silent /UNLOCKKEY:"(product unlock key)" [/log "{Log file path}"] [/InstallPath:{Location to install}]

```
[/InstallSamples:{true/false}] [/InstallAssemblies:{true/false}] [/UninstallExistAssemblies:{true/false}]
[/InstallToolbox:{true/false}]
```

Note: [...] – Arguments inside the square brackets are optional.

Example: “D:\Temp\syncfusionejs2x.x.x.x.exe” /Install silent /UNLOCKKEY:“product unlock key” /log
 “C:\Temp\EssentialStudioPlatform.log” /InstallPath:C:\Syncfusion\x.x.x.x /InstallSamples:true
 /InstallAssemblies:true /UninstallExistAssemblies:true /InstallToolbox:true

7. Essential Studio for JavaScript (Essential JS2) is installed.

Note: x.x.x.x should be replaced with the Essential Studio version and the Product Unlock Key needs to be replaced with the Unlock Key for that version.

Command Line Uninstallation

Syncfusion Essential JavaScript – EJ2 can be uninstalled silently using the Command Line.

1. Run the Syncfusion JavaScript – EJ2 installer by double-clicking it. The Installer Wizard automatically opens and extracts the package.
2. The syncfusionejs2_(version).exe file will be extracted into the Temp directory.
3. Run %temp%. The Temp folder will be opened. The syncfusionejs2_(version).exe file will be located in one of the folders.
4. Copy the extracted syncfusionejs2_(version).exe file in local drive.
5. Exit the Wizard.
6. Run Command Prompt in administrator mode and enter the following arguments.

Arguments: “Copied installer file path\ syncfusionejs2_(version).exe” /uninstall silent

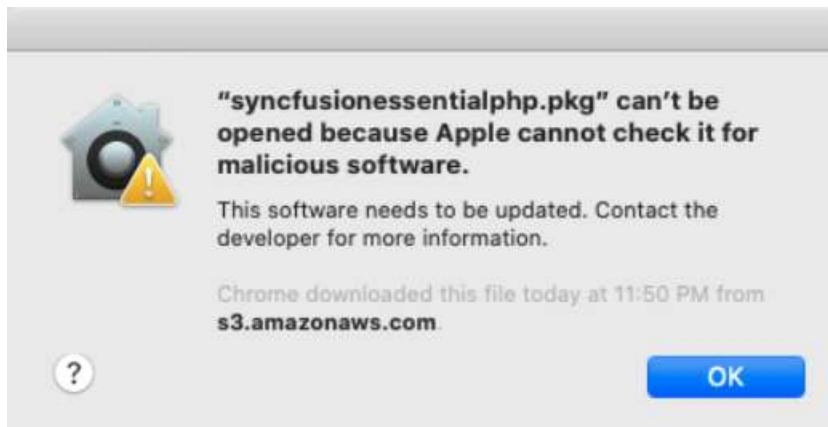
Example: “D:\Temp\syncfusionejs2_x.x.x.x.exe” /uninstall silent

7. Essential Studio for JavaScript (Essential JS2) is uninstalled.

Installing Syncfusion JavaScript – EJ2 Mac Installer

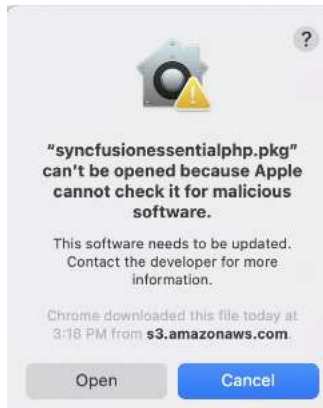
Steps to resolve the warning message in Catalina OS or later

While running Essential Studio JavaScript - EJ2 Mac Installer on Catalina MacOS or later, the below alert will be displayed.



If you receive this alert, follow the below steps for the easiest solution.

1. Right-click the downloaded pkg file.
2. Select the "Open With" option and choose "Installer (Default)". The following pop-up appears.

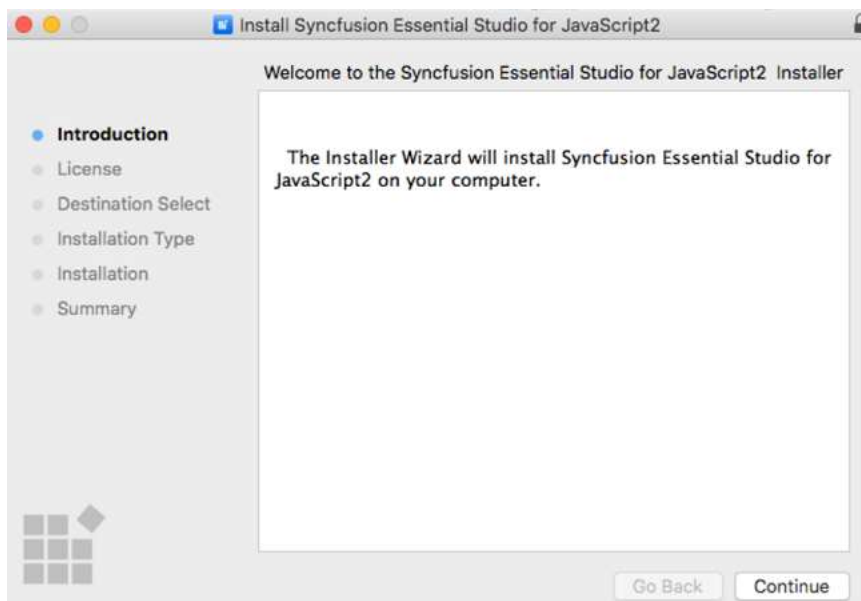


3. When you click "Open" the installer window will be opened.

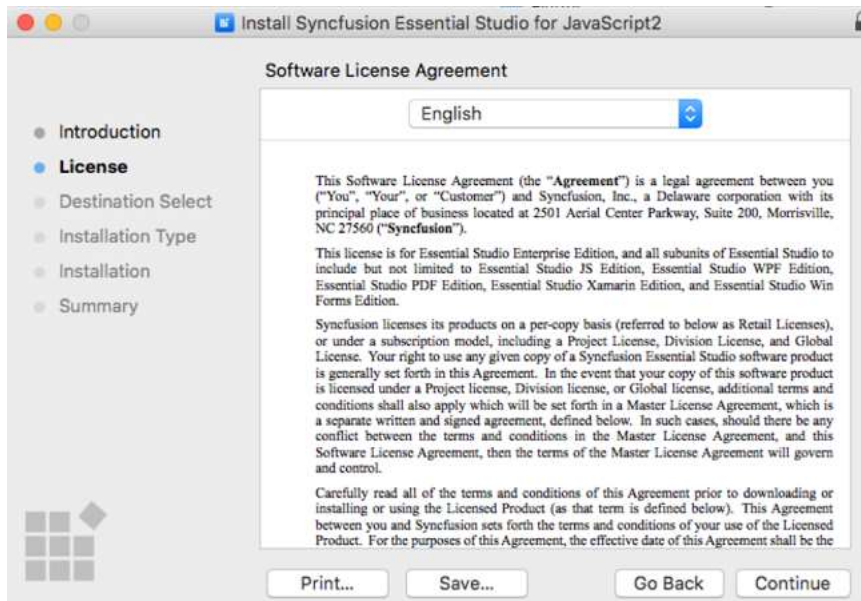
Step-by-Step Installation

The steps below show how to install the Essential Studio JavaScript - EJ2 Mac installer.

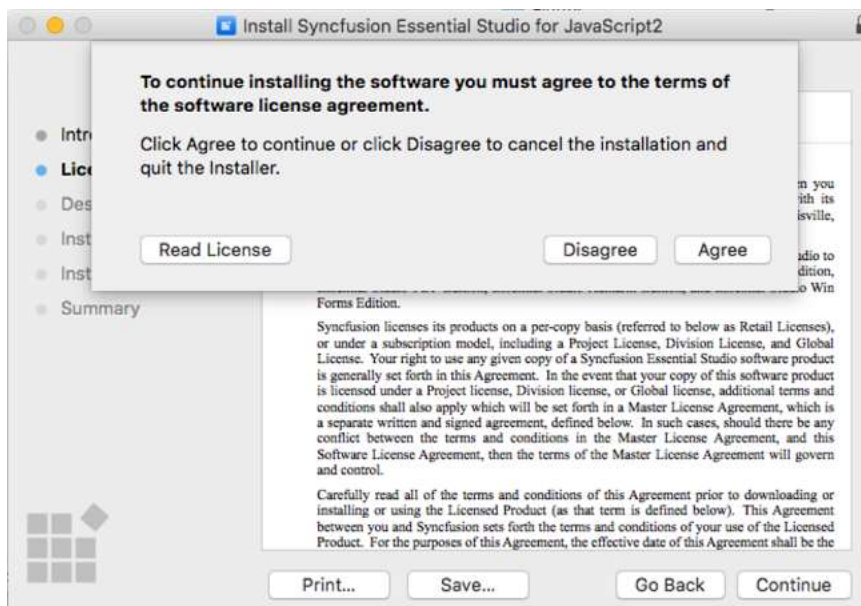
1. Double-click the Syncfusion Essential Studio JavaScript - EJ2 Mac installer(.pkg) file. The installer Wizard opens. Click Continue.



2. The Software License Agreement wizard will appear. Click the Continue button.

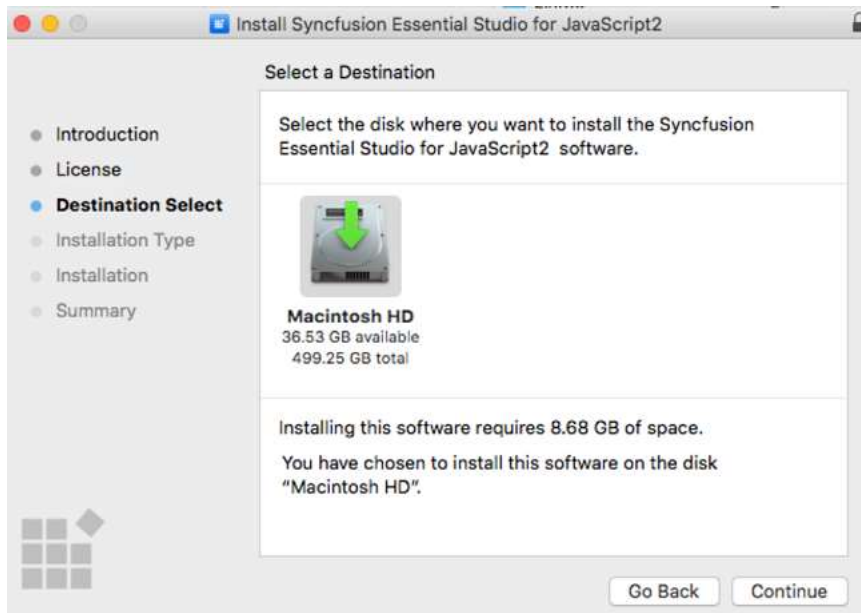


3. The License Agreement's Confirmation window will appear. If you have read the Software License Agreement, click **Agree**.

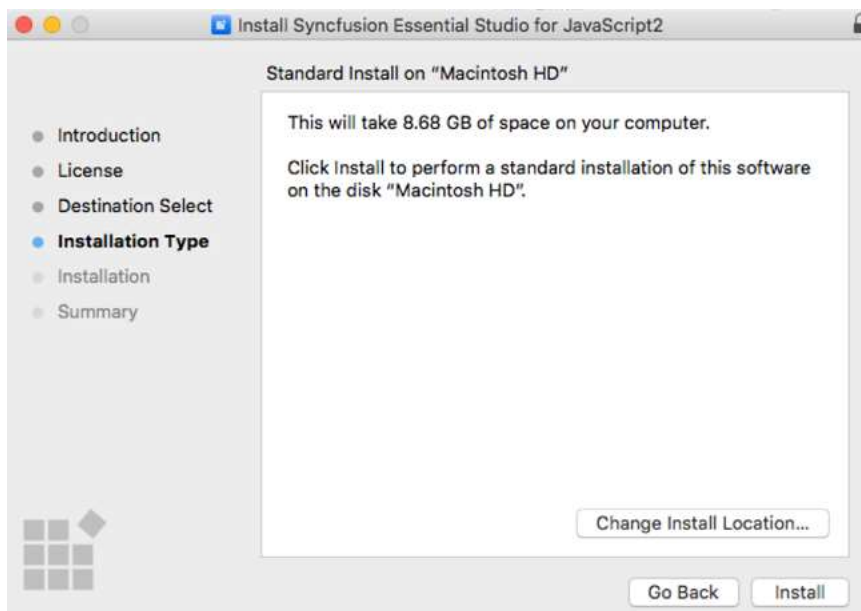


Note: The Unlock key is not required to install the Mac installer. The Syncfusion Essential Studio JavaScript - EJ2 Mac installer can be used for development purposes without registering the Unlock key.

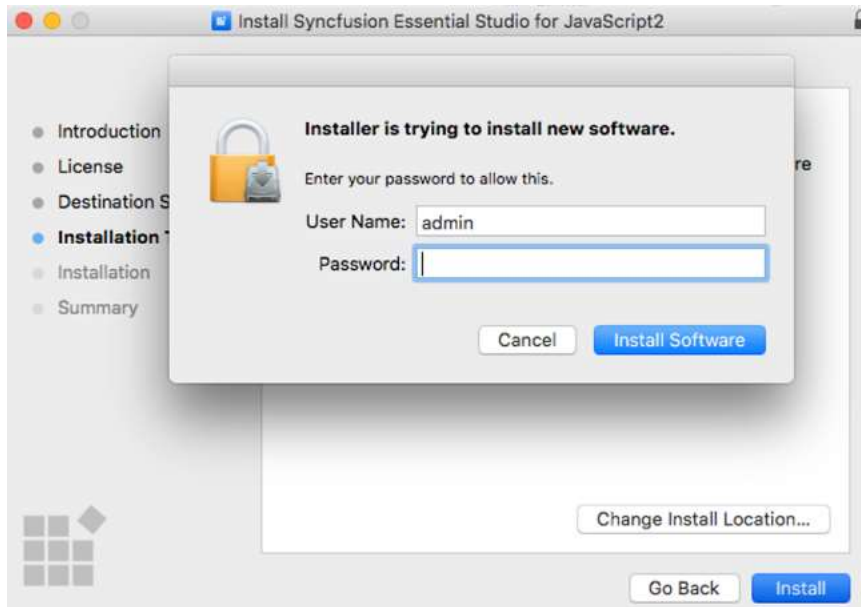
4. The Destination select wizard will appear. You can choose which disc to install the Syncfusion Essential Studio for JavaScript - EJ2 Mac installer on here.



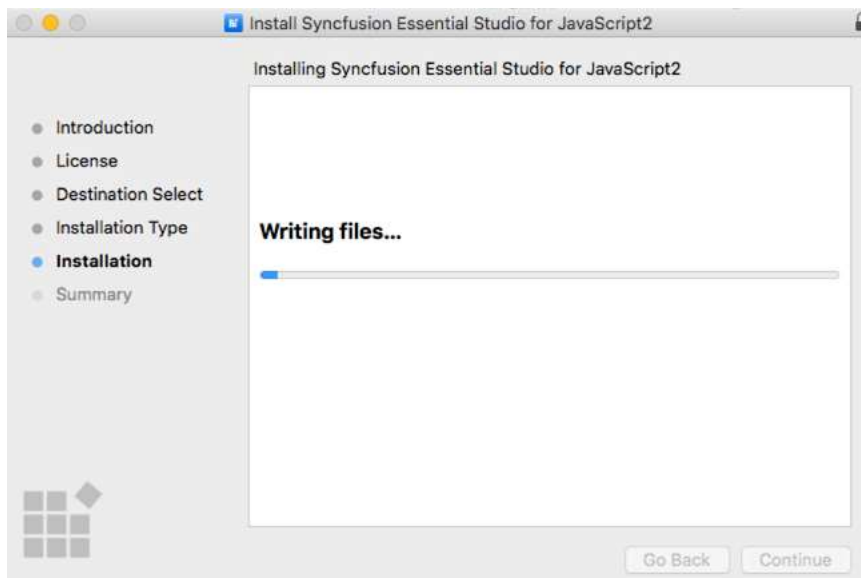
5. The Installation Type wizard will appear. Click Install to begin the standard installation of the Syncfusion Essential Studio JavaScript - EJ2 Mac installer.



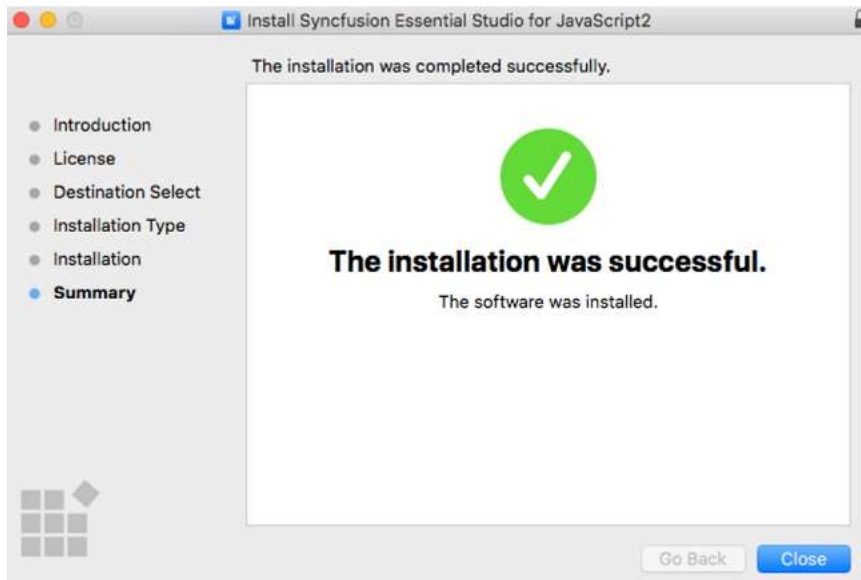
6. The Authentication window will appear. To begin the installation, enter the Mac machine's password and click **Install Software**.



7. The installation process will begin on your machine.

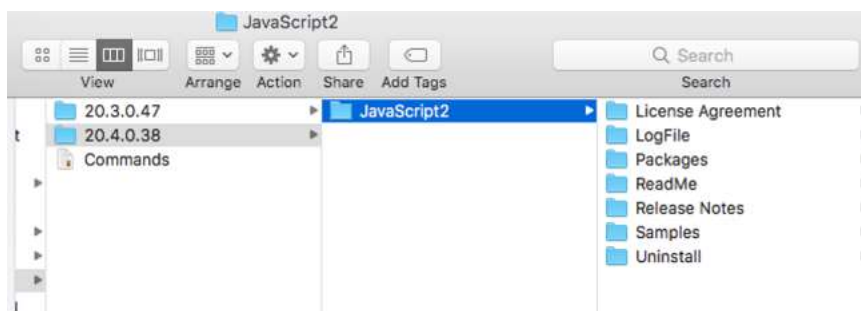


8. Once the installation is complete, the completed screen will be displayed. To exit the installation wizard, click Close.



By default, Mac installer will install the files in following location.

Location: {Documents}\Syncfusion\ {version}\ {platform}



License key registration in samples

After the installation, the license key is required to register the demo source that is included in the Mac installer. To learn about the steps for license registration for the JavaScript - EJ2 Mac installer, please refer to this.

Linux Installer

Download Syncfusion JavaScript Linux Installer

The Syncfusion installer can be downloaded from the [Syncfusion](https://www.syncfusion.com) website. You can either download the licensed installer or try our trial installer depending on your license.

- Trial Installer - Licensed Installer

You can download the Syncfusion installer from [Syncfusion.com](https://www.syncfusion.com) website

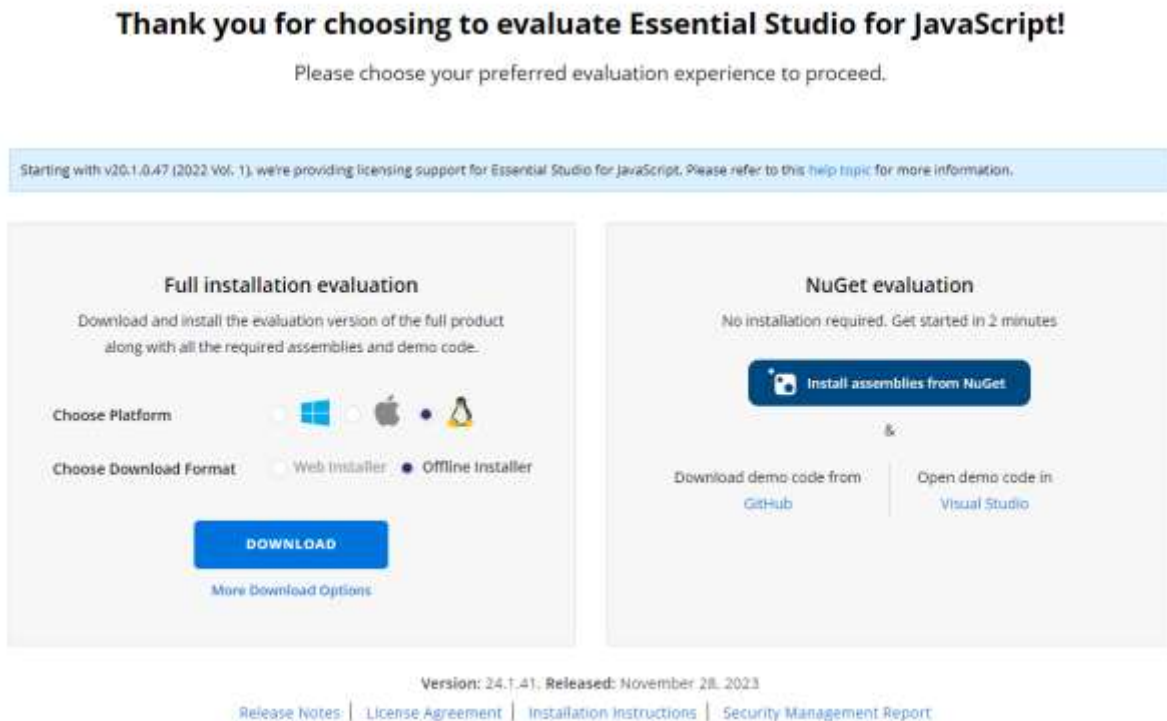
Download the Trial Version

Our 30-day trial can be downloaded in two ways.

- Download Free Trial Setup
- Start Trials if using components through [NuGet.org](https://www.nuget.org)

[Download Free Trial Setup](#)

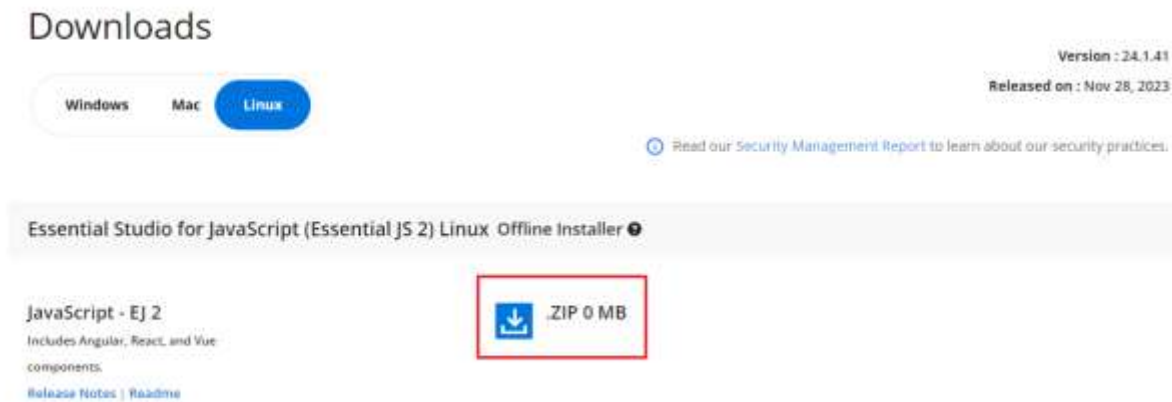
1. You can evaluate our 30-day free trial by visiting the [Download Free Trial](#) page and select the product
2. After completing the required form or logging in with your registered Syncfusion account, you can download the trial installer from the confirmation page. (as shown in below screenshot.)



3. With a trial license, only the latest version's trial installer can be downloaded.
4. Unlock key is not required to install the Syncfusion JavaScript Linux trial installer.
5. Before the trial expires, you can download the trial installer at any time from your registered account's [Trials & Downloads](#) page (as shown in below screenshot.)



6. Click the More Download Options (element 2 in the above screenshot) button to get the JavaScript Product Offline trial installer which is available in ZIP format.

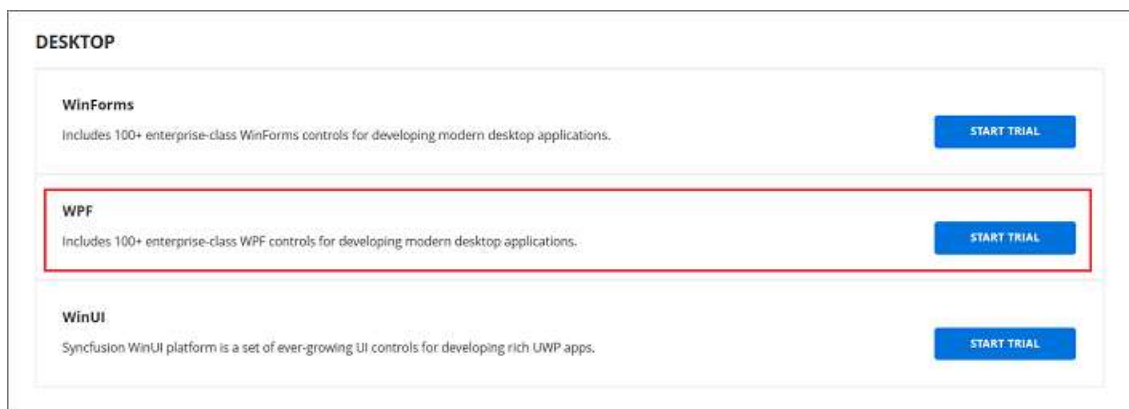


Start Trials if using components through [NuGet.org](https://www.nuget.org)

You should initiate an evaluation if you have already obtained our components through [NuGet.org](https://www.nuget.org/packages?q=syncfusion)

1. You can start your 30-day free trial from the [Start Trial](#) page from your account.

Note: You can generate the license key for your active trial products from [Trials & Downloads](#) page. This license key will be mandatory to use our trial products in your application. To know more about License key, refer this [help topic](#).



2. To access this page, you must sign up\log in with your Syncfusion account.
3. Begin your trial by selecting the Syncfusion product.

Note: If you've already used the trial products and they haven't expired, you won't be able to start the trial for the same product again.



5. You can find your current active trial products on the [Trials & Downloads](#) page.

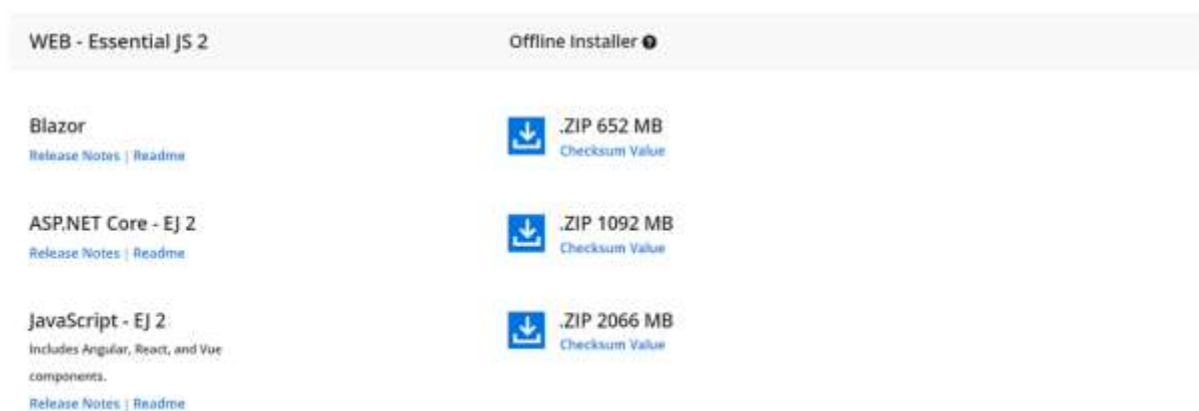
Download the License Version

1. Syncfusion licensed products will be available in the [License & Downloads](#) page under your registered Syncfusion account.
2. You can view all the licenses (both active and expired) associated with your account.
3. You can download JavaScript Linux licensed installer by going to More Downloads Options (element 3 in the screenshot below).



4. Unlock key is not required to install the Syncfusion JavaScript Linux trial installer.
5. For Linux OS, ZIP formats is available for download.

Downloads

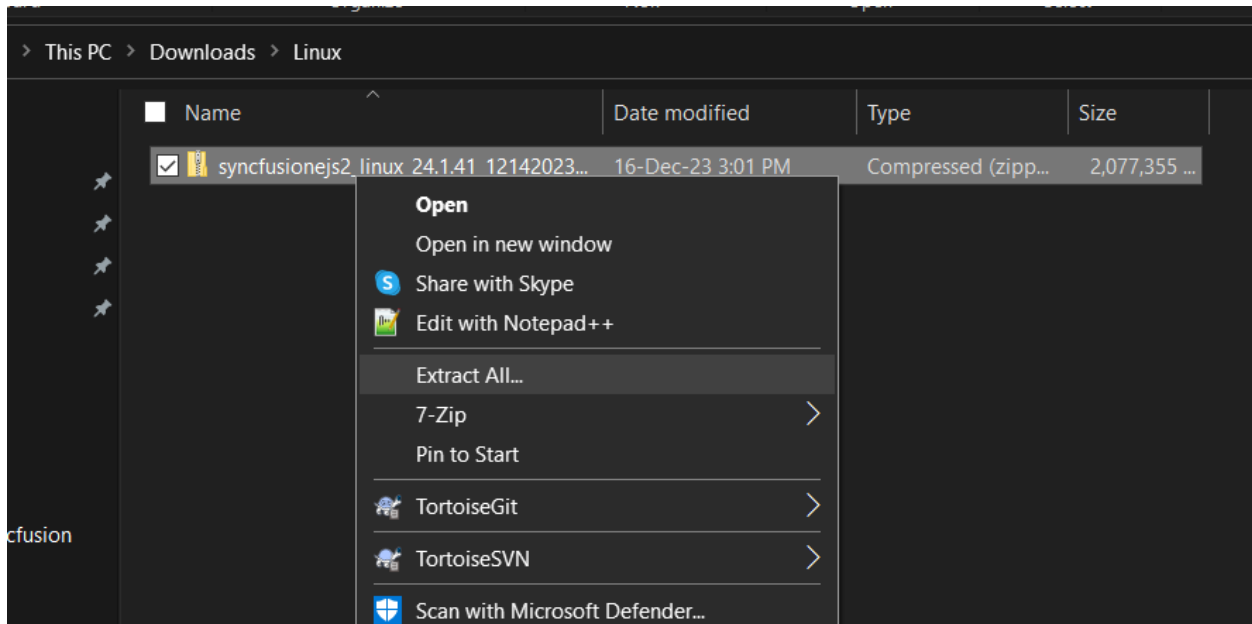


Installing Syncfusion JavaScript Linux installer

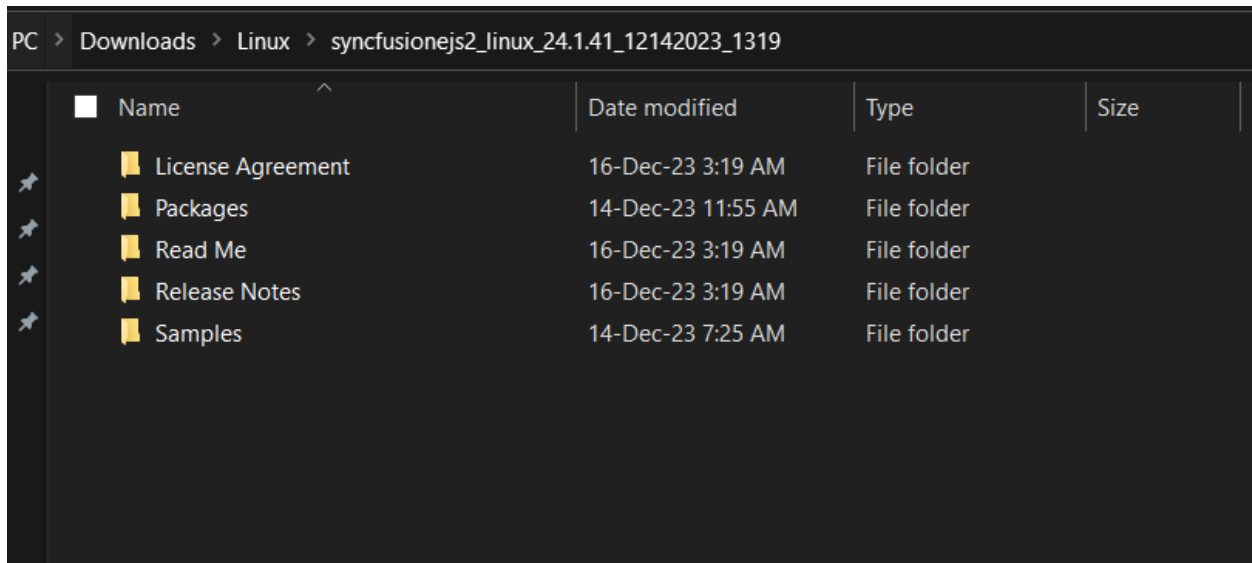
Step-by-Step Installation

The steps below show how to install JavaScript Linux installer.

1. Extract the Syncfusion JavaScript Linux installer(.zip) file. The files are extracted in your machine.



2. The Linux zip file contains the following folders.



Note: The Unlock key is not required to install the Linux installer.

4. You can launch the demo source and use the NuGet packages included in the Linux installer.

[License key registration in samples](#)

After the installation, the license key is required to register the demo source that is included in the Linux installer. To learn about the steps for license registration for the JavaScript - EJ2 Linux installer, please refer to this.

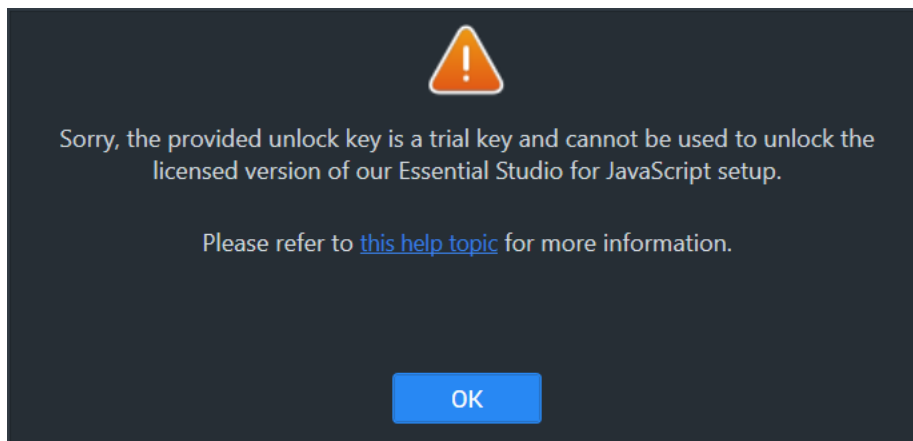
Common Installation Errors

This article describes the most common installation errors, as well as the causes and solutions to those errors.

- Unlocking the license installer using the trial key
- License has expired
- Unable to find a valid license or trial
- Unable to install because of another installation
- Unable to install due to controlled folder access

Unlocking the license installer using the trial key

Error Message: Sorry, the provided unlock key is a trial unlock key and cannot be used to unlock the licensed version of our Essential Studio for JavaScript installer.



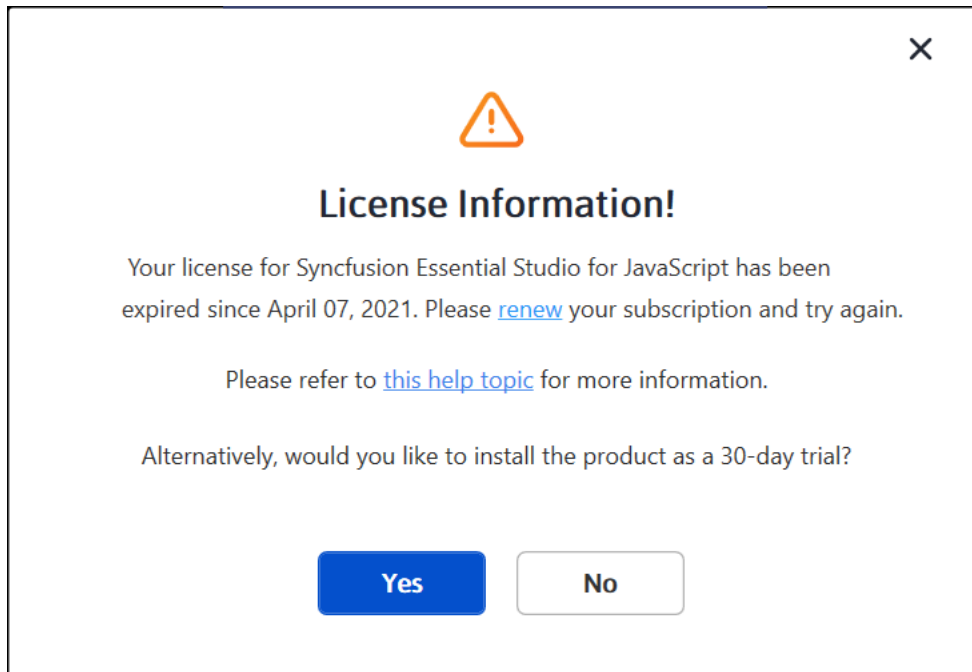
Reason
 You are attempting to use a Trial unlock key to unlock the licensed installer.

Suggested solution
 Only a licensed unlock key can unlock a licensed installer. So, to unlock the Licensed installer, use the Licensed unlock key. To generate the licensed unlock key, refer to [this](#) article.

License has expired

Error Message: Your license for Syncfusion Essential Studio for JavaScript – EJ2 has been expired since {date}. Please renew your subscription and try again.

Online Installer



Reason
 This error message will appear if your license has expired.

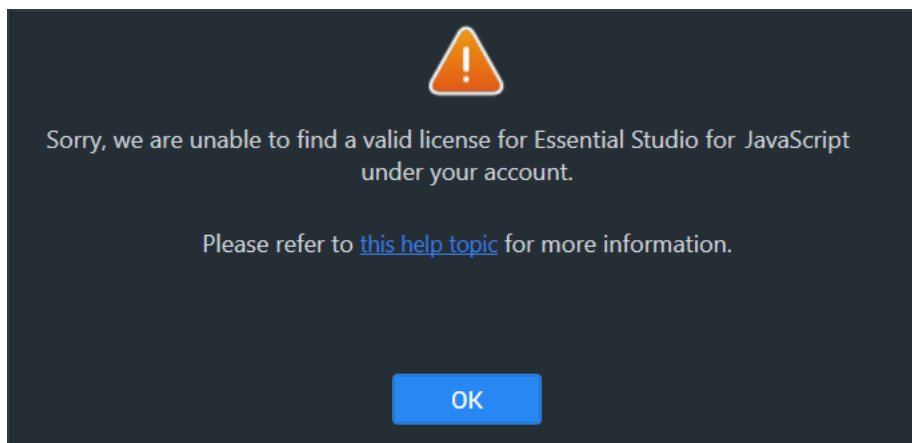
Suggested solution
 You can choose from the options below.

1. You can renew your subscription [here](#).
2. You can get a new license [here](#).
3. You can reach out to our sales team by emailing sales@syncfusion.com.
4. You can also extend the 30-day trial period after your trial license has expired.

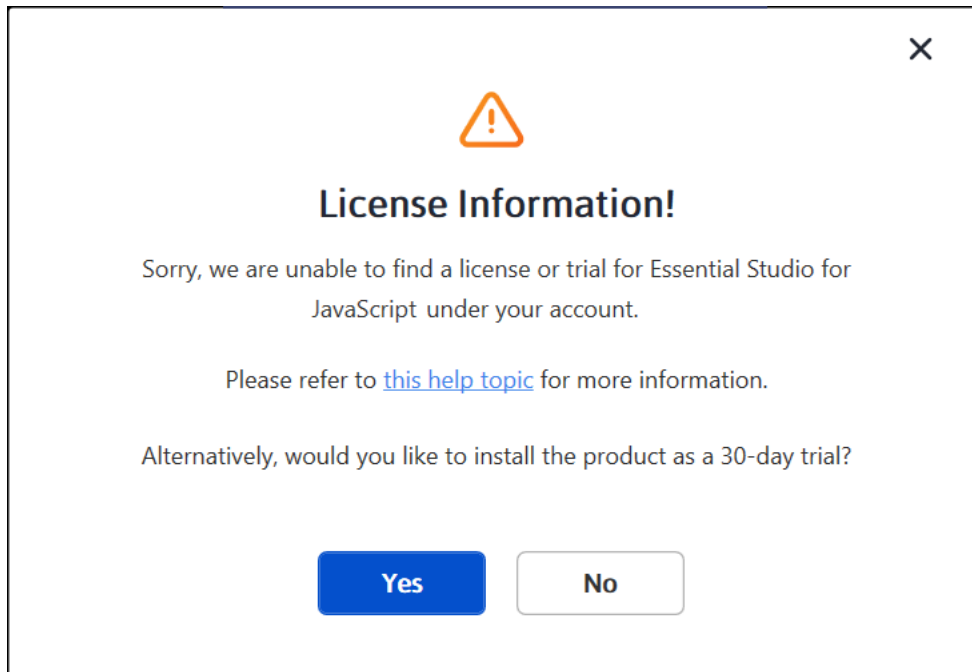
Unable to find a valid license or trial

Error Message: Sorry, we are unable to find a valid license or trial for Essential Studio for JavaScript – EJ2 under your account.

Offline installer



Online installer



Reason
 The following are possible causes of this error:

The following are possible causes of this error:

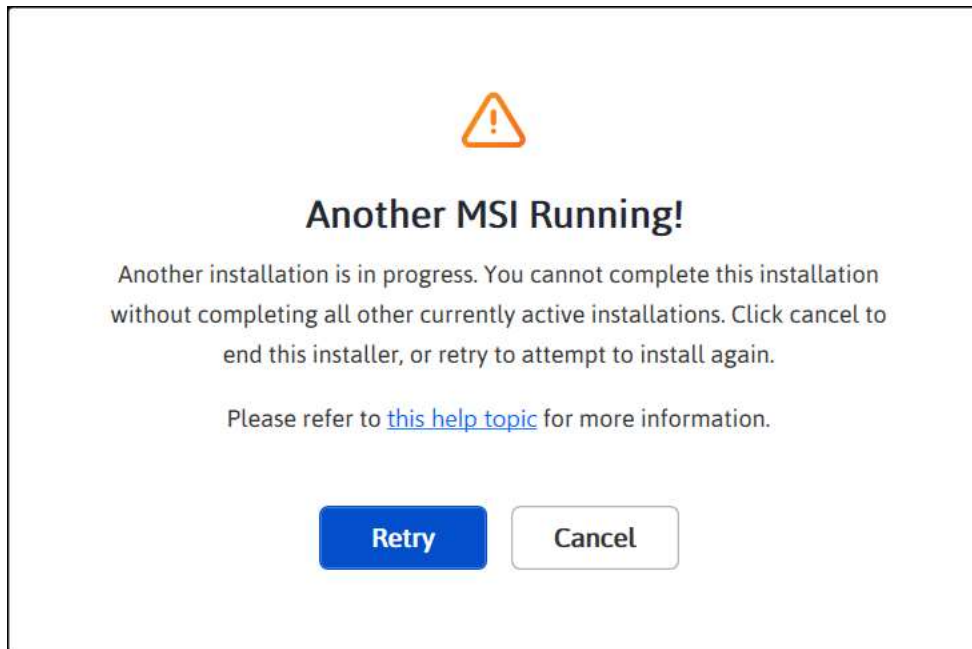
- When your trial period expired
- When you don't have a license or an active trial
- You are not the license holder of your license
- Your account administrator has not yet assigned you a license.

Suggested solution

1. You can get a new license [here](#).
2. Contact your account administrator.
3. Send an email to clientrelations@syncfusion.com to request a license.
4. You can reach out to our sales team by emailing sales@syncfusion.com.

Unable to install because of another installation

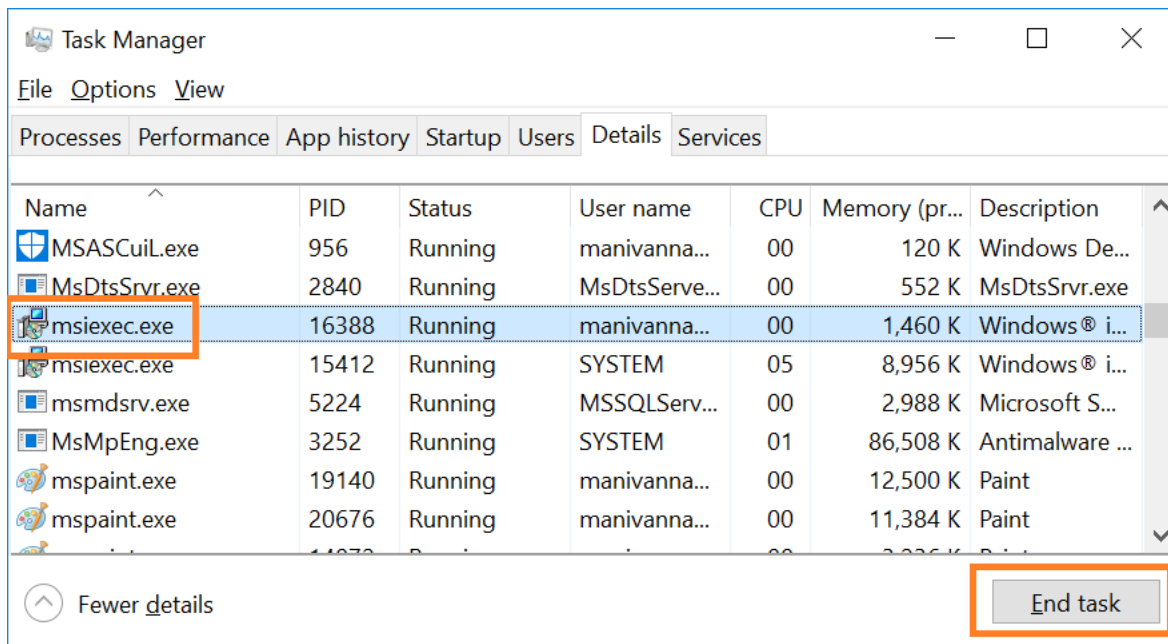
Error Message: Another installation is in progress. You cannot start this installation without completing all other currently active installations. Click cancel to end this installer or retry to attempt after currently active installation completed to install again.



Reason
 You are trying to install when another installation is already running in your machine.

Suggested solution
 Open and kill the msixec process in the task manager and then continue to install Syncfusion. If the problem is still present, restart the computer and try Syncfusion installer.

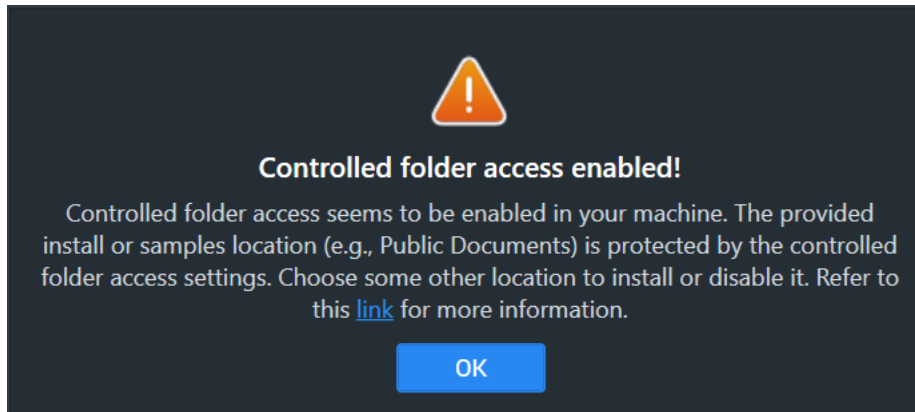
1. Open the Windows Task Manager.
2. Browse the Details tab.
3. Select the msixec.exe and click **End task**.



Unable to install due to controlled folder access

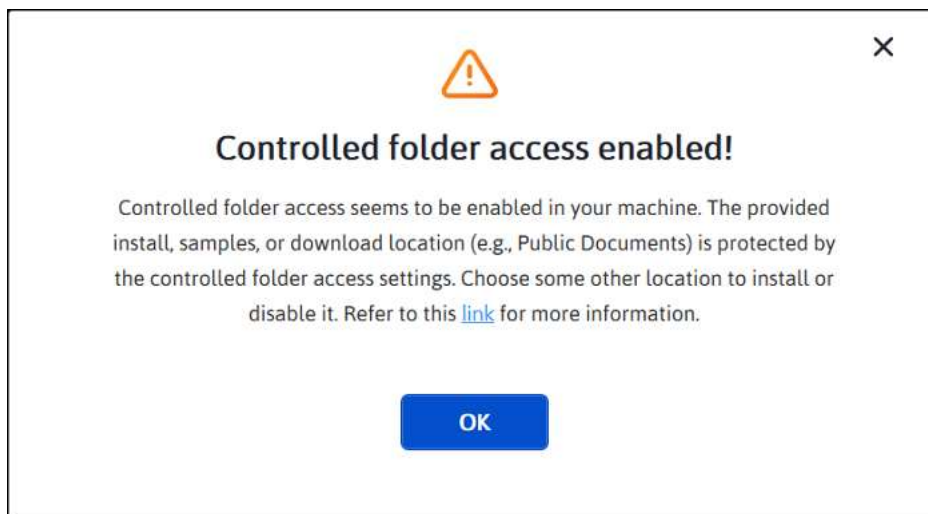
Offline

Error Message: Controlled folder access seems to be enabled in your machine. The provided install or samples location (e.g., Public Documents) is protected by the controlled folder access settings.



Online

Error Message: Controlled folder access seems to be enabled in your machine. The provided install, samples, or download location (e.g., Public Documents) is protected by the controlled folder access settings.



Reason
 You have enabled controlled folder access settings on your computer.

Suggested solution

Suggestion 1:

1. We will ship our demos in the public documents folder by default.
2. You have controlled folder access enabled on your machine, so our demos cannot be installed in the documents folder. If you need to install our demos in the Documents folder, follow the steps in this [link](#) and disable the controlled folder access.
3. You can enable this option after the installing our Syncfusion setup.

Suggestion 2:

1. If you do not want to disable controlled folder access, you can install our demos in another directory.

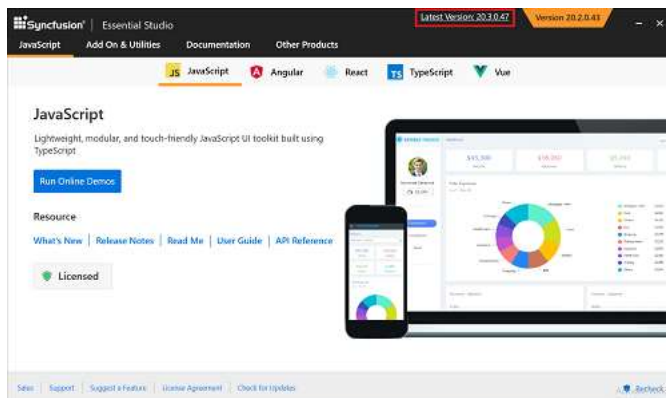
Upgrading Syncfusion JavaScript (Essential JS2)

Syncfusion releases new volumes once every three months, with exciting new features. There will be one Service Pack release for these volume releases. Service Pack releases are provided to address major bug fixes in the volume releases.

You can upgrade to our latest version from any installed Syncfusion version.

Upgrading to the latest version

The most recent version of Syncfusion JavaScript – EJ2 can be downloaded and installed by clicking on the “Latest Version: {Version}” link at the top of the Syncfusion JavaScript – EJ2 Control Panel.



You can also upgrade to the latest version just by downloading and installing the products you require from [this](#) link. The existing installed versions are not required to be uninstalled.

It is not required to install the Volume release before installing the Service Pack release. As releases for Volume and Service Packs work independently, you can install the latest version with major bug fixes directly.

Upgrade from trial version to license version

Uninstall the trial version and install the fully licensed installer from the [License and Downloads](#) section of our website to upgrade from the trial version.

Note: License key registration is not required for JavaScript, if you are using scripts (.js) and css files.

Licensing

Appearance

Theme in ##Platform_Name## controls

The Syncfusion JavaScript library has provided the below list of in-built themes:

|Theme | Style Sheet Name |

|-----|-----|

|Material 3 | material3.css |

|Material 3 Dark | material3-dark.css |

Fluent	fluent.css
Fluent Dark	fluent-dark.css
Bootstrap 5	bootstrap5.css
Bootstrap 5 Dark	bootstrap5-dark.css
Bootstrap 4	bootstrap4.css
Bootstrap 3	bootstrap.css
Bootstrap 3 Dark	bootstrap-dark.css
Google's Material	material.css
Google's Material-Dark	material-dark.css
Tailwind CSS	tailwind.css
Tailwind Dark CSS	tailwind-dark.css
Microsoft Office Fabric	fabric.css
Microsoft Office Fabric Dark	fabric-dark.css
High Contrast	highcontrast.css

The Syncfusion Bootstrap theme is designed based on **Bootstrap v3**, but it can be compatible with **Bootstrap v4** applications. In addition to these four built-in themes, [ThemeStudio](#) also provides support for the Fusion theme, which can only be downloaded from [ThemeStudio](#).

Reference themes in the application

Syncfusion JavaScript controls themes that can be used in the application by referencing the style sheet. Using the following approaches, themes can be referenced in the application.

- [NPM Packages](#) - Used to customize the existing themes and bundle stylesheet's in an application.
- [Content Delivery Network \(CDN\)](#) - Used to reference complete css via static web assets.
- [Theme Studio](#) - Used to customize and generate themes only for the selected (used) components.

NPM packages

All Syncfusion JavaScript (Essential JS 2) packages are available on the [npmjs.com](#) public registry. Themes are shipped as individual and combined CSS files. A combined CSS file can be referred to from the npm package [@syncfusion/ej2](#) and individual CSS files are available within the same control repository's `style` folder. In the `ej2` npm package, we have shipped both CSS and SCSS files for all controls.

Referring all controls theme

Referring all control CSS theme from ej2 package

,

```
@import "../nodemodules/@syncfusion/ej2/<themename>.css";
```

,

Referring all control SCSS theme from ej2 package

```
`scss
@import "../nodemodules/@syncfusion/ej2/<themenam>.scss";
`
```

Referring individual control theme

The individual control theme can be referred from the [individual package](#) or from the **ej2** package.

Referring individual control SCSS theme from an individual package.

```
`scss
@import "<dependent-package>/<dependent-control>/<theme_name>.scss";
@import "ej2-buttons/styles/button/<theme_name>.scss";
`
```

Example:

```
`scss
@import "ej2-base/styles/material.scss";
@import "ej2-buttons/styles/button/material.scss";
`
```

ej2-base is common dependent package for all Syncfusion JavaScript control styles. so, it needs to be added first in the import statement.

Referring individual control SCSS theme from ej2 package

```
`scss
@import "ej2/<dependent-control>/<theme_name>.scss";
@import "ej2/button/<theme_name>.scss";
`
```

Example:

```
`scss
@import "ej2/base/material.scss";
@import "ej2/button/material.scss";
`
```

CDN reference

Syncfusion hosts every Syncfusion JavaScript control as a separate package on the CDN. This allows you to load the scripts and styles for each individual package. Syncfusion also provides a single package that includes all Syncfusion JavaScript controls, allowing you to load the scripts and styles for all controls as a single script and style file.

Single CDN theme reference for all controls

Refer to a single CDN link that contains all Syncfusion JavaScript control styles as follows:

`https://cdn.syncfusion.com/ej2/<version>/<theme_name>.css`

Theme Name	CDN Reference
---	---
Material 3	https://cdn.syncfusion.com/ej2/22.1.34/material3.css
Material 3 Dark	https://cdn.syncfusion.com/ej2/22.1.34/material3-dark.css
Fluent	https://cdn.syncfusion.com/ej2/22.1.34/fluent.css
Fluent Dark	https://cdn.syncfusion.com/ej2/22.1.34/fluent-dark.css
Bootstrap 5	https://cdn.syncfusion.com/ej2/22.1.34/bootstrap5.css
Bootstrap 5 Dark	https://cdn.syncfusion.com/ej2/22.1.34/bootstrap5-dark.css
Bootstrap 4	https://cdn.syncfusion.com/ej2/22.1.34/bootstrap4.css
Bootstrap 3	https://cdn.syncfusion.com/ej2/22.1.34/bootstrap.css
Bootstrap 3 Dark	https://cdn.syncfusion.com/ej2/22.1.34/bootstrap-dark.css
Google's Material	https://cdn.syncfusion.com/ej2/22.1.34/material.css
Google's Material Dark	https://cdn.syncfusion.com/ej2/22.1.34/material-dark.css
Tailwind CSS	https://cdn.syncfusion.com/ej2/22.1.34/tailwind.css
Tailwind CSS Dark	https://cdn.syncfusion.com/ej2/22.1.34/tailwind-dark.css
Microsoft Office Fabric	https://cdn.syncfusion.com/ej2/22.1.34/fabric.css
Microsoft Office Fabric Dark	https://cdn.syncfusion.com/ej2/22.1.34/fabric-dark.css
High Contrast	https://cdn.syncfusion.com/ej2/22.1.34/highcontrast.css

Individual control theme CDN reference

The primary goal of individual CDN control is to optimize the loading time and memory of the website or app in the production stage. Individual control package loading should be done in the order indicated by its dependency graph. The CDN of the dependency packages should be manually included before the intended individual control package CDN.

Dependency style:

```
https://cdn.syncfusion.com/ej2/22.1.34/{DEPENDENCYPACKAGE_NAME}/styles/material.css
```

Control style:

```
https://cdn.syncfusion.com/ej2/22.1.34/{PACKAGE_NAME}/styles/material.css
```

Example:

Button's control style:

<https://cdn.syncfusion.com/ej2/22.1.34/ej2-buttons/styles/material.css>

Button's dependency style:

<https://cdn.syncfusion.com/ej2/22.1.34/ej2-base/styles/material.css>

Change theme dynamically

In the application, Syncfusion themes can be changed dynamically by changing their style reference as follows.

- Add the theme CDN link and DropDownList element to the HTML file using the following code.

```
`html
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<meta name="author" content="Syncfusion JavaScript Grid control" />
<!-- Syncfusion JavaScript controls styles -->
<link id="css-link" href="https://cdn.syncfusion.com/ej2/22.1.34/bootstrap5.css" rel="stylesheet" />
</head>
<body>
<div id='container' style="margin: 50px;">
<div id='dropdown' style="margin:0 auto auto 0; width:300px; padding-bottom: 20px;">
<!--element which is going to render the DropDownList-->
<input type="text" tabindex="1" id='themes-dropdown' />
</div>
</div>
</body>
</html>
`
```

- The following code example demonstrates how to change the theme dynamically in the application using Syncfusion JavaScript DropDownList control.

```

`ts
import { DropDownList, ChangeEventArgs } from '@syncfusion/ej2-dropdowns';

// define the array of themes data
let themesData: string[] = ['material3', 'material3-dark', 'bootstrap5', 'bootstrap5-dark', 'fluent', 'fluent-dark', 'material', 'tailwind', 'tailwind-dark', 'fabric', 'fabric-dark'];

// initialize DropDownList component
let dropDownListObject: DropDownList = new DropDownList({
//set the data to dataSource property
dataSource: themesData,
placeholder: "Choose a theme",
change: (args: ChangeEventArgs) => {
let themeName = args.value as string;
document.getElementsByTagName('body')[0].style.display = 'none';
let styleLink: any = document.getElementById('css-link');
styleLink.href = 'https://cdn.syncfusion.com/ej2/22.1.34/' + themeName + '.css';
setTimeout(function () { document.getElementsByTagName('body')[0].style.display = 'block'; }, 250);
}
});

// render initialized DropDownList
dropDownListObject.appendTo('#themes-dropdown');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="author" content="Syncfusion JavaScript Grid control">
  <!-- Syncfusion JavaScript controls styles -->
  <link id="css-link"
href="https://cdn.syncfusion.com/ej2/24.1.41/bootstrap5.css"
rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin: 50px;">

```

```

<div id="dropdown" style="margin:0 auto auto 0; width:300px;
padding-bottom: 20px;">
    <!--element which is going to render the DropDownList-->
    <input type="text" tabindex="1" id="themes-dropdown">
</div>
<!--element which is going to render-->
<div id="Grid"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Common variables

The following list of common variables are used in the Syncfusion JavaScript library themes for all UI controls. You can change these variables to customize the corresponding theme.

<!-- markdownlint-disable MD033 -->

Syncfusion Material 3 theme

Name	Value (Default Theme)	Value (Dark Theme)
--color-sf-black	rgb(0,0,0)	rgb(0,0,0)
--color-sf-white	rgb(255,255,255)	rgb(255,255,255)
--color-sf-primary	rgb(103, 80, 164)	rgb(208, 188, 255)
--color-sf-primary-container	rgb(234, 221, 255)	rgb(79, 55, 139)
--color-sf-on-primary	rgb(255, 255, 255)	rgb(55, 30, 115)
--color-sf-on-primary-container	rgb(33, 0, 94)	rgb(234, 221, 255)
--color-sf-surface	rgb(255, 255, 255)	rgb(28, 27, 31)
--color-sf-surface-variant	rgb(231, 224, 236)	rgb(73, 69, 79)
--color-sf-on-surface	rgb(28, 27, 31)	rgb(230, 225, 229)
--color-sf-on-surface-variant	rgb(73, 69, 78)	rgb(202, 196, 208)
--color-sf-secondary	rgb(98, 91, 113)	rgb(204, 194, 220)
--color-sf-secondary-container	rgb(232, 222, 248)	rgb(74, 68, 88)

Name	Value (Default Theme)	Value (Dark Theme)
--color-sf-on-secondary	rgb(255, 255, 255)	rgb(51, 45, 65)
--color-sf-on-secondary-container	rgb(30, 25, 43)	rgb(232, 222, 248)
--color-sf-tertiary	rgb(125, 82, 96)	rgb(239, 184, 200)
--color-sf-tertiary-container	rgb(255, 216, 228)	rgb(99, 59, 72)
--color-sf-on-tertiary	rgb(255, 255, 255)	rgb(73, 37, 50)
--color-sf-on-tertiary-containe	rgb(55, 11, 30)	rgb(255, 216, 228)
--color-sf-background	rgb(255, 255, 255)	rgb(28, 27, 31)
--color-sf-on-background	rgb(28, 27, 31)	rgb(230, 225, 229)
--color-sf-outline	rgb(121, 116, 126)	rgb(147, 143, 153)
--color-sf-outline-variant	rgb(196, 199, 197)	rgb(68, 71, 70)
--color-sf-shadow	rgb(0, 0, 0)	rgb(0, 0, 0)
--color-sf-surface-tint-color	rgb(103, 80, 164)	rgb(208, 188, 255)
--color-sf-inverse-surface	rgb(49, 48, 51)	rgb(230, 225, 229)
--color-sf-inverse-on-surface	rgb(244, 239, 244)	rgb(49, 48, 51)
--color-sf-inverse-primary	rgb(208, 188, 255)	rgb(103, 80, 164)
--color-sf-scrim	rgb(0, 0, 0)	rgb(0, 0, 0)
--color-sf-error	rgb(179, 38, 30)	rgb(242, 184, 181)
--color-sf-error-container	rgb(249, 222, 220)	rgb(140, 29, 24)
--color-sf-on-error	rgb(255, 250, 250)	rgb(96, 20, 16)
--color-sf-on-error-container	rgb(65, 14, 11)	rgb(249, 222, 220)
--color-sf-success	rgb(32, 81, 7)	rgb(83, 202, 23)
--color-sf-success-container	rgb(209, 255, 186)	rgb(22, 62, 2)

Name	Value (Default Theme)	Value (Dark Theme)
--color-sf-on-success	rgb(244, 255, 239)	rgb(13, 39, 0)
--color-sf-on-success-container	rgb(13, 39, 0)	rgb(183, 250, 150)
--color-sf-info	rgb(1, 87, 155)	rgb(71, 172, 251)
--color-sf-info-container	rgb(233, 245, 255)	rgb(0, 67, 120)
--color-sf-on-info	rgb(250, 253, 255)	rgb(0, 51, 91)
--color-sf-on-info-container	rgb(0, 51, 91)	rgb(173, 219, 255)
--color-sf-warning	rgb(145, 76, 0)	rgb(245, 180, 130)
--color-sf-warning-container	rgb(254, 236, 222)	rgb(123, 65, 0)
--color-sf-on-warning	rgb(255, 255, 255)	rgb(99, 52, 0)
--color-sf-on-warning-container	rgb(47, 21, 0)	rgb(255, 220, 193)

Syncfusion Bootstrap 5 theme

Name	Value (Default Theme)	Value (Dark Theme)
\$black	#000	#000
\$white	#fff	#fff
\$gray-100	#f8f9fa	#f8f9fa
\$gray-200	#e9ecef	#e9ecef
\$gray-300	#dee2e6	#dee2e6
\$gray-400	#ced4da	#ced4da
\$gray-500	#adb5bd	#adb5bd
\$gray-600	#6c757d	#6c757d
\$gray-700	#495057	#495057
\$gray-800	#343a40	#343a40

Name	Value (Default Theme)	Value (Dark Theme)
\$gray-900	#212529	#212529
\$blue	#0d6efd	#0d6efd
\$indigo	#6610f2	#6610f2
\$purple	#6f42c1	#6f42c1
\$pink	#d63384	#d63384
\$red	#dc3545	#dc3545
\$orange	#fd7e14	#fd7e14
\$yellow	#ffc107	#ffc107
\$green	#198754	#198754
\$teal	#20c997	#20c997
\$cyan	#0dcaf0	#0dcaf0

Syncfusion Fluent theme

Name	Value (Default Theme)	Value (Dark Theme)
\$black	#000	#000
\$white	#fff	#fff
\$gray220	#11100f	#11100f
\$gray210	#161514	#161514
\$gray200	#1b1a19	#1b1a19
\$gray190	#201f1e	#201f1e
\$gray180	#252423	#252423
\$gray170	#292827	#292827
\$gray160	#323130	#323130

Name	Value (Default Theme)	Value (Dark Theme)
\$gray150	#3b3a39	#3b3a39
\$gray140	#484644	#484644
\$gray130	#605e5c	#605e5c
\$gray120	#797775	#797775
\$gray110	#8a8886	#8a8886
\$gray100	#979593	#979593
\$gray90	#a19f9d	#a19f9d
\$gray80	#b3b0ad	#b3b0ad
\$gray70	#bebbb8	#bebbb8
\$gray60	#c8c6c4	#c8c6c4
\$gray50	#d2d0ce	#d2d0ce
\$gray40	#e1dfdd	#e1dfdd
\$gray30	#edebe9	#edebe9
\$gray20	#f3f2f1	#f3f2f1
\$gray10	#faf9f8	#faf9f8
\$cyanblue10	#0078d4	#0078d4
\$red10	#d13438	#d13438
\$orange20	#ca5010	#ca5010
\$green20	#0b6a0b	#0b6a0b
\$cyan20	#038387	#038387

Syncfusion Bootstrap 4 theme

Name	Value
\$white	#fff
\$gray-100	#f8f9fa
\$gray-200	#e9ecef
\$gray-300	#dee2e6
\$gray-400	#ced4da
\$gray-500	#adb5bd
\$gray-600	#6c757d
\$gray-700	#495057
\$gray-800	#343a40
\$gray-900	#212529
\$black	#000
\$blue	#007bff
\$indigo	#6610f2
\$purple	#6f42c1
\$pink	#e83e8c
\$red	#dc3545
\$orange	#fd7e14
\$yellow	#ffc107
\$green	#28a745
\$teal	#20c997
\$cyan	#17a2b8

Syncfusion Bootstrap theme

Name	Value (Default Theme)	Value (Dark Theme)
\$brand-primary	#317ab9	#0070f0
\$brand-primary-darken-10	#3071a9	darken(\$brand-primary, 10%)
\$brand-primary-darken-15	#2a6496	darken(\$brand-primary, 20%)
\$brand-primary-darken-25	#1f496e	darken(\$brand-primary, 30%)
\$brand-primary-darken-35	#142f46	darken(\$brand-primary, 40%)
\$brand-primary-font	#fff	#fff
\$gray-base	#000	#1a1a1a
\$gray-darker	#222	#131313
\$gray-dark	#333	#2a2a2a
\$gray	#555	#313131
\$gray-light	#777	#393939
\$grey-44	#444	#414141
\$grey-88	#888	#484848
\$grey-99	#999	#505050
\$grey-8c	#8c8c8c	#585858
\$grey-ad	#adadad	#676767
\$grey-dark-font	#fff	#f0f0f0
\$grey-white	#fff	#6e6e6e
\$grey-lighter	#eee	#767676
\$grey-f9	#f9f9f9	#7e7e7e
\$grey-f8	#f8f8f8	#858585

Name	Value (Default Theme)	Value (Dark Theme)
\$grey-f5	#f5f5f5	#8d8d8d
\$grey-e6	#e6e6e6	#959595
\$grey-dd	#ddd	#9c9c9c
\$grey-d4	#d4d4d4	#a4a4a4
\$grey-cc	#ccc	#acacac
\$grey-light-font	#333	#fff
\$brand-success	#5cb85c	#48b14c
\$brand-success-dark	#3c763d	#358238
\$brand-info	#5bc0de	#2aaac0
\$brand-info-dark	#31708f	#208090
\$brand-warning	#f0ad4e	#fac168
\$brand-warning-dark	#8a6d3b	#f9ad37
\$brand-danger	#d9534f	#d44f4f
\$brand-danger-dark	#a94442	#c12f2f
\$brand-success-light	#dff0d8	#dff0d8
\$brand-info-light	#d9edf7	#d9edf7
\$brand-warning-light	#fcf8e3	#fcf8e3
\$brand-danger-light	#f2dede	#f2dede
\$input-border-focus	#66afe9	#104888
\$brand-success-font	#3c763d	#2f7432
\$brand-info-font	#31708f	#1a6c7a
\$brand-warning-font	#8a6d3b	#9d6106

Name	Value (Default Theme)	Value (Dark Theme)
\$brand-danger-font	#a94442	#ac2a2a
\$base-font	#000	#000
\$brand-primary-lighten-10		lighten(\$brand-primary, 10%)
\$brand-primary-lighten-15		lighten(\$brand-primary, 15%)
\$brand-primary-lighten-20		lighten(\$brand-primary, 20%)
\$brand-primary-lighten-30		lighten(\$brand-primary, 30%)
\$brand-primary-lighten-40		lighten(\$brand-primary, 40%)

Syncfusion Material theme

Name	Value (Default Theme)	Value (Dark Theme)
\$accent	#e3165b	#ff80ab
\$accent-font	#fff	#000
\$primary	#3f51b5	#3f51b5
\$primary-50	#e8eaf6	#e8eaf6
\$primary-100	#c5cae9	#c5cae9
\$primary-200	#9fa8da	#9fa8da
\$primary-300	#7986cb	#7986cb
\$primary-font	#fff	#fff
\$primary-50-font	#000	#000
\$primary-100-font	#000	#000
\$primary-200-font	#000	#000
\$primary-300-font	#fff	#fff
\$grey-white	#fff	#fff

Name	Value (Default Theme)	Value (Dark Theme)
\$grey-black	#000	#000
\$grey-50	#fafafa	#fafafa
\$grey-100	#f5f5f5	#f5f5f5
\$grey-200	#eee	#eee
\$grey-300	#e0e0e0	#e0e0e0
\$grey-400	#bdbdbd	#bdbdbd
\$grey-500	#9e9e9e	#9e9e9e
\$grey-600	#757575	#757575
\$grey-700	#616161	#616161
\$grey-800	#424242	#424242
\$grey-900	#212121	#212121
\$grey-dark	#303030	#303030
\$grey-light-font	#000	#000
\$grey-dark-font	#fff	#fff
\$base-font	#000	#000
\$error-font	#f44336	#ff6652
\$success-bg		#4caf50
\$error-bg		#ff6652
\$warning-bg		#ff9800
\$info-bg		#03a9f4
\$message-font		#fff
\$success-font		#4caf50

Name	Value (Default Theme)	Value (Dark Theme)
\$warning-font		#ff9800
\$info-font		#03a9f4

Syncfusion Microsoft Office Fabric theme

Name	Value (Default Theme)	Value (Dark Theme)
\$theme-primary	#0078d6	#0074cc
\$theme-dark-alt	darken(\$theme-primary, 3%)	darken(\$theme-primary, 3%)
\$theme-dark	darken(\$theme-primary, 10%)	darken(\$theme-primary, 6%)
\$theme-darker	darken(\$theme-primary, 18%)	darken(\$theme-primary, 10%)
\$theme-secondary	lighten(\$theme-primary, 3%)	lighten(\$theme-primary, 3%)
\$theme-tertiary	lighten(\$theme-primary, 21%)	lighten(\$theme-primary, 21%)
\$theme-light	lighten(\$theme-primary, 44%)	lighten(\$theme-primary, 44%)
\$theme-lighter	lighten(\$theme-primary, 49%)	lighten(\$theme-primary, 49%)
\$theme-lighter-alt	lighten(\$theme-primary, 55%)	lighten(\$theme-primary, 55%)
\$neutral-white	#fff	#201f1f
\$neutral-lighter-alt	#f8f8f8	#282727
\$neutral-lighter	#f4f4f4	#333232
\$neutral-light	#eaeaea	#414040
\$neutral-quintenaryalt	#dadada	#4a4848
\$neutral-quintenary	#d0d0d0	#514f4f
\$neutral-tertiary-alt	#c8c8c8	#6f6c6c
\$neutral-tertiary	#a6a6a6	#9a9a9a
\$neutral-secondary-alt	#767676	#c8c8c8

Name	Value (Default Theme)	Value (Dark Theme)
\$neutral-secondary	#666	#dadada
\$neutral-primary	#333	#fff
\$neutral-dark	#212121	#f4f4f4
\$neutral-black	#000	#f8f8f8
\$alert-bg	#deecf9	#bf7500
\$error-bg	#fde7e9	#cd2a19
\$success-bg	#dff6dd	#37844d
\$theme-dark-font	#fff	#fff
\$theme-primary-font	#fff	#fff
\$theme-light-font	#333	#000
\$neutral-light-font	#333	#dadada
\$neutral-light-fontalt	#000	#fff
\$grey-dark-font	#fff	#000
\$base-font	#333	#dadada
\$message-font	#333	#fff
\$alert-font	#d83b01	#ff9d48
\$error-font	#a80000	#ff5f5f
\$success-font	#107c10	#8eff8d
\$info-bg		#1e79cb
\$info-font		#62cfff

Syncfusion High Contrast theme

Name	Value
\$selection-bg	#ffd939
\$selection-font	#000
\$selection-border	#ffd939
\$hover-bg	#685708
\$hover-font	#fff
\$hover-border	#fff
\$border-default	#969696
\$border-alt	#757575
\$border-fg	#fff
\$border-fg-alt	#ffd939
\$bg-base-0	#000
\$bg-base-5	#0d0d0d
\$bg-base-10	#1a1a1a
\$bg-base-15	#262626
\$bg-base-20	#333
\$bg-base-75	#bfbfbf
\$bg-base-100	#fff
\$header-font	#ffd939
\$header-font-alt	#fff
\$content-font	#fff
\$content-font-alt	#969696

Name	Value
\$link	#8a8aff
\$invert-font	#000
\$success-bg	#166600
\$error-bg	#b30900
\$message-font	#fff
\$alert-bg	#944000
\$info-bg	#0056b3
\$success-alt	#2bc700
\$error-alt	#ff6161
\$alert-alt	#ff7d1a
\$info-alt	#66b0ff
\$disable	#757575

Syncfusion Tailwind CSS theme

Name	Value (Default Theme)	Value (Dark Theme)
\$black	#000	#000
\$white	#fff	#fff
\$transparent	transparent	transparent
\$cool-gray-50	#f9fafb	#f9fafb
\$cool-gray-100	#f3f4f6	#f3f4f6
\$cool-gray-200	#e5e7eb	#e5e7eb
\$cool-gray-300	#d1d5db	#d1d5db
\$cool-gray-400	#9ca3af	#9ca3af

Name	Value (Default Theme)	Value (Dark Theme)
\$cool-gray-500	#6b7280	#6b7280
\$cool-gray-600	#4b5563	#4b5563
\$cool-gray-700	#374151	#374151
\$cool-gray-800	#1f2937	#1f2937
\$cool-gray-900	#111827	#111827
\$red-100	#fee2e2	#fee2e2
\$red-400	#f87171	#f87171
\$red-500	#ef4444	#ef4444
\$red-600	#dc2626	#dc2626
\$red-800	#991b1b	#991b1b
\$green-100	#dcfce7	#dcfce7
\$green-500	#22c55e	#22c55e
\$green-600	#16a34a	#16a34a
\$green-700	#15803d	#15803d
\$orange-100	#ffedd5	#ffedd5
\$orange-500	#f97316	#f97316
\$orange-600	#ea580c	#ea580c
\$orange-700	#c2410c	#c2410c
\$orange-800	#9a3412	#9a3412
\$cyan-300	#67e8f9	#67e8f9
\$cyan-400	#22d3ee	#22d3ee
\$cyan-500	#06b6d4	#06b6d4

Name	Value (Default Theme)	Value (Dark Theme)
\$cyan-600	#0891b2	#0891b2
\$cyan-800	#155e75	#155e75
\$indigo-50	#eef2ff	
\$indigo-100	#e0e7ff	
\$indigo-200	#c7d2fe	
\$indigo-300	#a5b4fc	
\$indigo-400	#818cf8	
\$indigo-500	#6366f1	
\$indigo-600	#4f46e5	
\$indigo-700	#4338ca	
\$indigo-800	#3730a3	
\$indigo-900	#312e81	
\$green-400		#4ade80
\$light-blue-50		#f0f9ff
\$light-blue-100		#e0f2fe
\$light-blue-400		#38bdf8
\$light-blue-500		#0ea5e9
\$light-blue-600		#0284c7
\$light-blue-700		#0369a1
\$light-blue-800		#075985

Size modes Size Mode for Syncfusion ##Platform_Name## Controls

An application that is designed to be accessed through a web browser on various devices, including desktop computers and mobile devices, may have a distinct layout or user interface on a mobile device compared to a desktop computer to better suit the smaller screen size.

Syncfusion JavaScript controls support both touch (bigger) and normal size modes. Touch mode creates a responsive design for mobile devices by adding the `e-bigger` class, which enhances interactions, visibility, and the overall experience.

Size mode for application

The user can enable touch mode (bigger) for the entire application by adding the `e-bigger` class to the `body` element in the `index.html` file as follows:

```
<body className="e-bigger">
```

```
...
```

```
</body>
```

Size mode for a control

The user can enable touch mode (bigger) for a control by adding the `e-bigger` class to the `div` element that contains the control. Another way of enabling touch mode is by adding the `e-bigger` class using the available `cssClass` property of the control.

INDEX.JS

```
// initialize button control
var button = new ej.buttons.Button({ content: 'Button' });
button.appendTo('#element')
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container" class="e-bigger">
```

```

        <button id="element">Button</button>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Change size mode for application at runtime

The user can change the size mode of the application between touch and normal (mouse) mode at runtime by adding and removing the `e-bigger` class. The following steps explain how to change the size mode of an application at runtime:

INDEX.JS

```

var touchButton = new ej.buttons.Button({ content: 'Touch Mode' });
touchButton.appendTo('#touch')
touchButton.element.onclick = () => {
    document.body.classList.add('e-bigger');
}
// initialize mouse button control
var mouseButton = new ej.buttons.Button({ content: 'Mouse Mode' });
mouseButton.appendTo('#mouse')
mouseButton.element.onclick = () => {
    document.body.classList.remove('e-bigger');
}
// initialize Calendar component
var calendar = new ej.calendars.Calendar();
calendar.appendTo('#calendar')

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>

```



```

</head>
<body>

  <div id="container">
    <div>
      <button id="touch"></button>
      <button id="mouse"></button>
    </div>
    <div class="control">
      <div id="calendar"></div>
    </div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Change size mode for a control at runtime

The user can change the size mode of a control between touch and normal (mouse) mode at runtime by setting the `e-bigger` CSS class.

INDEX.JS

```

var touchButton = new ej.buttons.Button({ content: 'Touch Mode' });
touchButton.appendTo('#touch')
touchButton.element.onclick = () => {
  var controls = document.querySelectorAll('.control');
  for (var index = 0; index < controls.length; index++) {
    controls[index].classList.add('e-bigger');
  }
}
// initialize mouse button control
var mouseButton = new ej.buttons.Button({ content: 'Mouse Mode' });
mouseButton.appendTo('#mouse')
mouseButton.element.onclick = () => {
  var controls = document.querySelectorAll('.control');
  for (var index = 0; index < controls.length; index++) {
    controls[index].classList.remove('e-bigger');
  }
}
// initialize Calendar component
var calendar = new ej.calendars.Calendar();
calendar.appendTo('#calendar')

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">

```

```

<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <button id="touch"></button>
            <button id="mouse"></button>
        </div>
        <div class="control">
            <div id="calendar"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See also

- [Sidebar responsiveness](#)
- [DataGrid responsiveness](#)
- [TreeGrid responsiveness](#)
- [Dashboard Layout responsiveness](#)
- [Kanban responsiveness](#)
- [Toolbar responsiveness](#)
- [Tab responsiveness](#)

Icons Library

Syncfusion's icon library is a collection of pre-designed icons that can be used to enhance the user interface of an application. This pre-designed icons are set of `base64` formatted font icons. Utilizing this icon library can make it simpler to create a cohesive, visually pleasing design for an application.

Referring icons in JavaScript application

Using the below approaches, the icons can be referenced in the JavaScript application.

- [npm package](#) - Use the npm package to access icons.
- [CDN reference](#) - Use the static web asset to access icons.

The npm package

All Syncfusion theme icons are shipped in the [ej2-icons](#) package, which is published on the [npmjs.com](#) public registry. This package contains both CSS and SCSS theme files for all themes.

Icons can be used from the npm package `ej2-icons`. To use the icons, install the npm package using the following command:

```
`bash
npm install @syncfusion/ej2-icons
```

Refer to the following syntax to use icons in a JavaScript application:

```
@import "../nodemodules/@syncfusion/ej2-icons/<themename>.css";
```

Example:

```
@import "../node_modules/@syncfusion/ej2-icons/material.css";
```

CDN reference

All Syncfusion theme icons are available on the CDN. Instead of using a local resource on the server, use a cloud CDN to refer to the icons.

Make sure that the version of the icons in the URL matches the version of the Syncfusion React package. This will prevent compatibility issues and ensure that the correct version of the icons is loaded.

To use the icons from the CDN, refer to the icons by URLs in the application. This can be done by linking the icons in the HTML file by adding a link tag to the head section.

```
// Bootstrap5
<head>
<link href="https://cdn.syncfusion.com/ej2/22.1.34/ej2-icons/styles/bootstrap5.css" rel="stylesheet"/>
</head>

//Material
<head>
<link href="https://cdn.syncfusion.com/ej2/22.1.34/ej2-icons/styles/material.css" rel="stylesheet"/>
</head>
```

Steps to use icons library

Let's create a JavaScript application using the following command:

For an introduction and configuration of the common specifications, see [getting started with the Syncfusion JavaScript application](#).

Using icons directly in HTML element

The built-in Syncfusion icons can be rendered directly in the HTML element by defining the `e-icons` class, which contains the font-family and common properties of font icons, and defining the available icon's class with the `e-` prefix.

The following steps explain the direct rendering of the Syncfusion icon in the HTML element.

1. Add the class name `e-icons` to the HTML element that needs to render the icon. 2. Add the icon class with corresponding icon content from the [available icons](#). For example, the below code snippet represents the paste icon class.

```
,
.e-paste:before {
content: '\e355';
}
,
```

3. Add `e-icons` and `e-paste` classes to the HTML element.

```
,
<span class="e-icons e-paste"></span>
,
```

4. Add the CDN link reference of icons library in the `~index.html` file.

```
,
<link href="https://cdn.syncfusion.com/ej2/22.1.34/ej2-icons/styles/bootstrap5.css" rel="stylesheet" />
,
```

Icon size

The `ej2-icons` package offers options to display icons in different size modes. A user can use different icon sizes in their application based on touch or mouse mode. If the user is using touch mode, add `e-large` class to the element to make the icon easily interactable, or add the `e-small` or `e-medium` class in mouse mode.

The pre-defined icon size is present in the available classes listed below.

- `e-small` - Sets the icon size as 8px.
- `e-medium` - Sets the icon size to 16px.
- `e-large` - Sets the icon size to 24px.

Example:

```
,
```

```
<span class="e-icons e-small e-search"></span>
```

```
<span class="e-icons e-medium e-search"></span>
```

```
<span class="e-icons e-large e-search"></span>
```

```
,
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container">

    <div>
      <p><b>Smaller Icons</b></p>
      <div class="small-icon-bar">
        <span class="e-icons e-small e-cut"></span>
        <span class="e-icons e-small e-copy"></span>
        <span class="e-icons e-small e-paste"></span>
      </div>
      <p><b>Medium Icons</b></p>
      <div class="medium-icon-bar">
        <span class="e-icons e-medium e-cut"></span>
        <span class="e-icons e-medium e-copy"></span>
        <span class="e-icons e-medium e-paste"></span>
      </div>
      <p><b>Larger Icons</b></p>
      <div class="large-icon-bar">
        <span class="e-icons e-large e-cut"></span>
        <span class="e-icons e-large e-copy"></span>
        <span class="e-icons e-large e-paste"></span>
      </div>
    </div>
  </div>
</body>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Icon appearance customization

The Syncfusion JavaScript icons can be customized with custom color and size by overriding the `e-icons` class. Customizing the icons in the library can be useful for making the icons more visually appealing and fitting to the design of the application. For example, a user can change the color of an icon to match the color scheme of their application, or increase the size of an icon to make it more visible on smaller screens. It may also be useful for creating a consistent look and feel across different parts of the application. Overall, customizing the icons in the library can improve the overall user experience of the application.

In the example below, the icon colour is customized with custom color.

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>
  <div id="container">

    <div class="icon-bar">
      <span class="e-icons e-cut"></span>
      <span class="e-icons e-copy"></span>
      <span class="e-icons e-paste"></span>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Available icons

The complete package of Essential JS 2 icons is listed below. The corresponding icon content can be referred in the content section.

```
<!-- markdownlint-disable MD033 -->
```

Material

```
<iframe class="doc-sample-frame"
src="https://ej2.syncfusion.com/products/icons/material/demo.html"
style="height:1000px;width:100%;"></iframe>
```

Fabric

```
<iframe class="doc-sample-frame" src="https://ej2.syncfusion.com/products/icons/fabric/demo.html"
style="height:1000px;width:100%;"></iframe>
```

Bootstrap

```
<iframe class="doc-sample-frame"
src="https://ej2.syncfusion.com/products/icons/bootstrap/demo.html"
style="height:1000px;width:100%;"></iframe>
```

Bootstrap 4

```
<iframe class="doc-sample-frame"
src="https://ej2.syncfusion.com/products/icons/bootstrap4/demo.html"
style="height:1000px;width:100%;"></iframe>
```

Bootstrap 5

```
<iframe class="doc-sample-frame"
src="https://ej2.syncfusion.com/products/icons/bootstrap5/demo.html"
style="height:1000px;width:100%;"></iframe>
```

High Contrast

```
<iframe class="doc-sample-frame"
src="https://ej2.syncfusion.com/products/icons/highcontrast/demo.html"
style="height:1000px;width:100%;"></iframe>
```

Tailwind CSS

```
<iframe class="doc-sample-frame"
src="https://ej2.syncfusion.com/products/icons/tailwind/demo.html"
style="height:1000px;width:100%;"></iframe>
```

Fluent

```
<iframe class="doc-sample-frame" src="https://ej2.syncfusion.com/products/icons/fluent/demo.html"
style="height:1000px;width:100%;"></iframe>
```

Overview

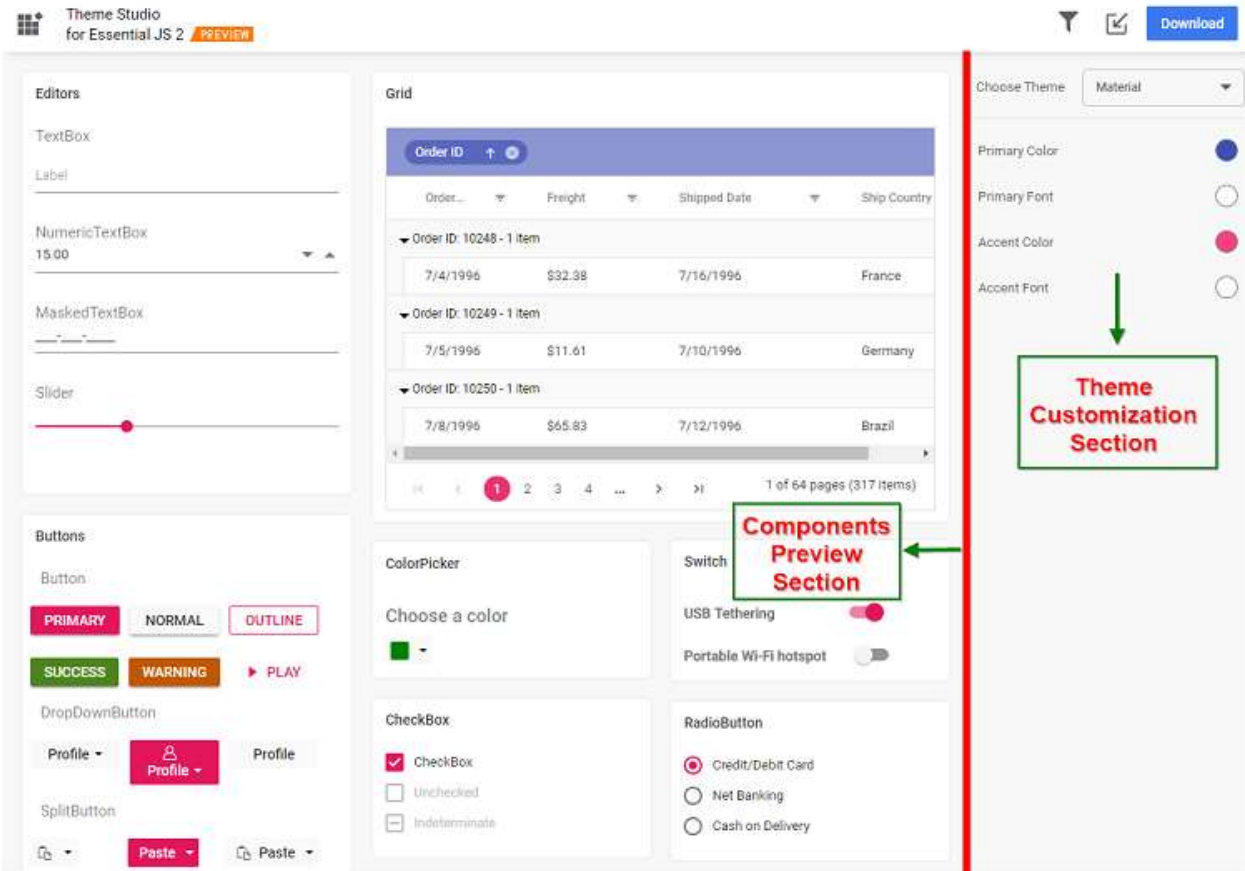
Theme Studio for Essential JS 2 can be used to customize a new theme from an existing theme. It doesn't support with Data visualization controls like Chart, Diagram, Gauge, Range Navigator, Maps.

Customizing theme color from theme studio

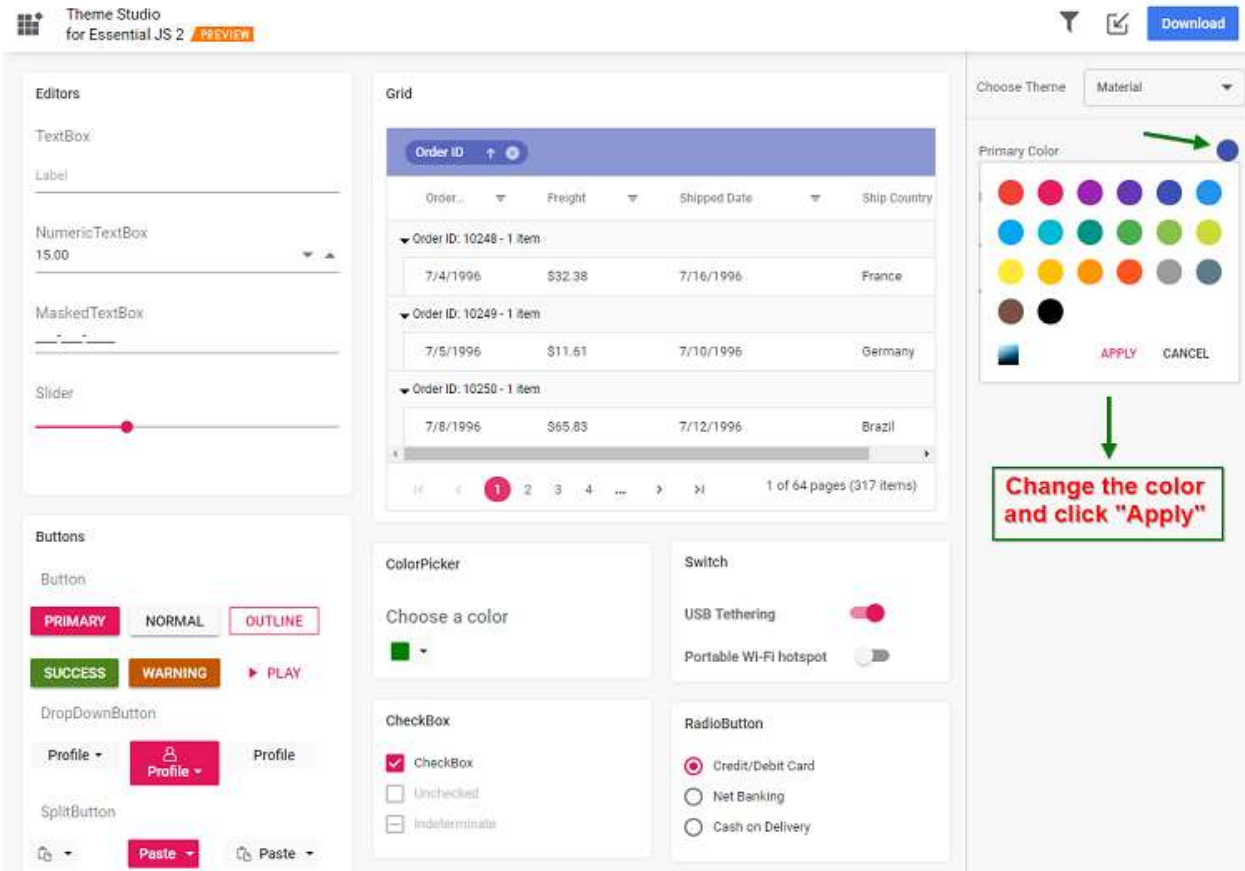
The Essential JS 2 themes are developed under the SCSS environment. Each theme has a unique common variable list. When you change the common variable color code value, it will reflect in all the Syncfusion JavaScript controls. All Syncfusion JavaScript control styles are derived from these [theme-based common variables](#). This common variable list is handled inside the theme studio application for customizing theme-based colors.

Step 1: Navigate to the theme studio application at <https://ej2.syncfusion.com/themestudio/>.

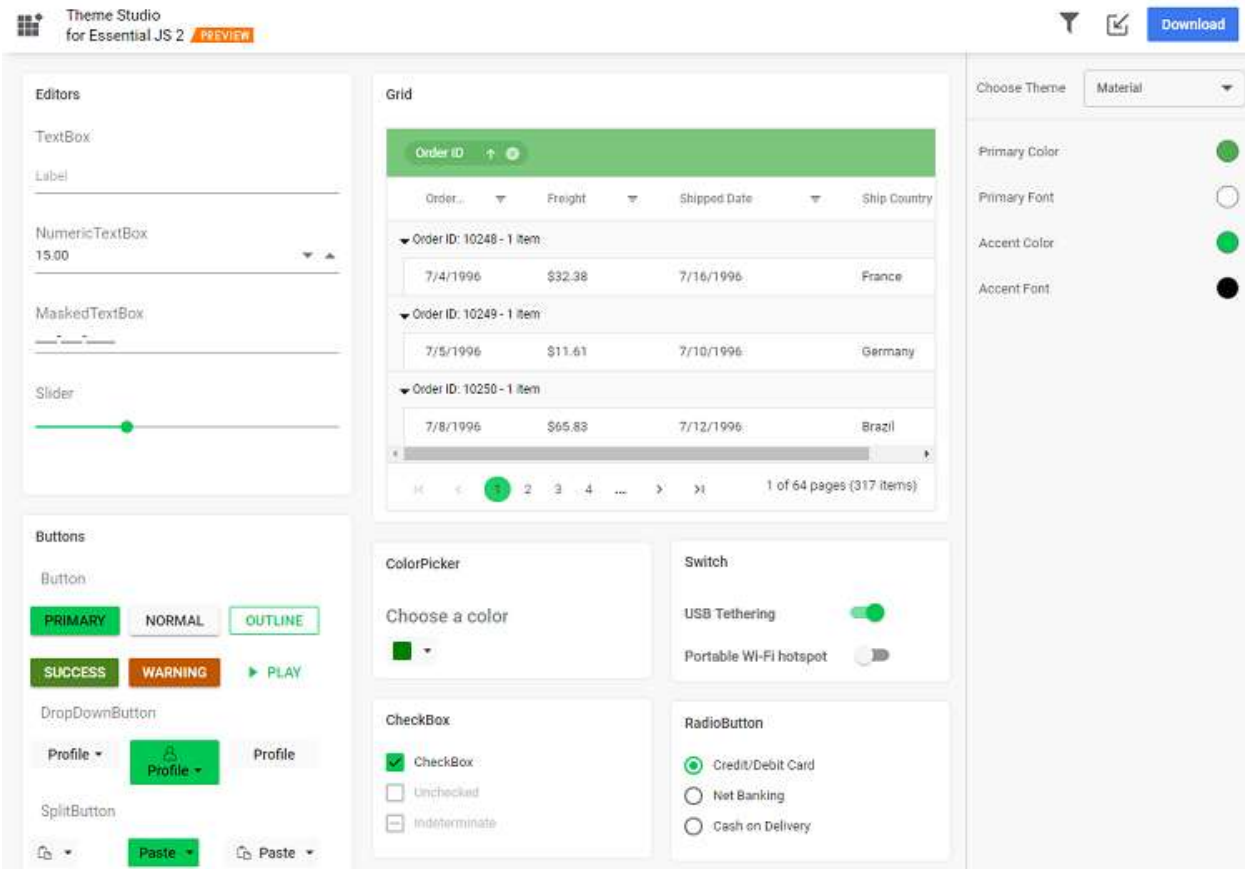
Step 2: The theme studio application page can be divided into two sections: the controls preview section on the left, and the theme customization section on the right.



Step 3: Click the color pickers in the theme customization section to select your desired colors.



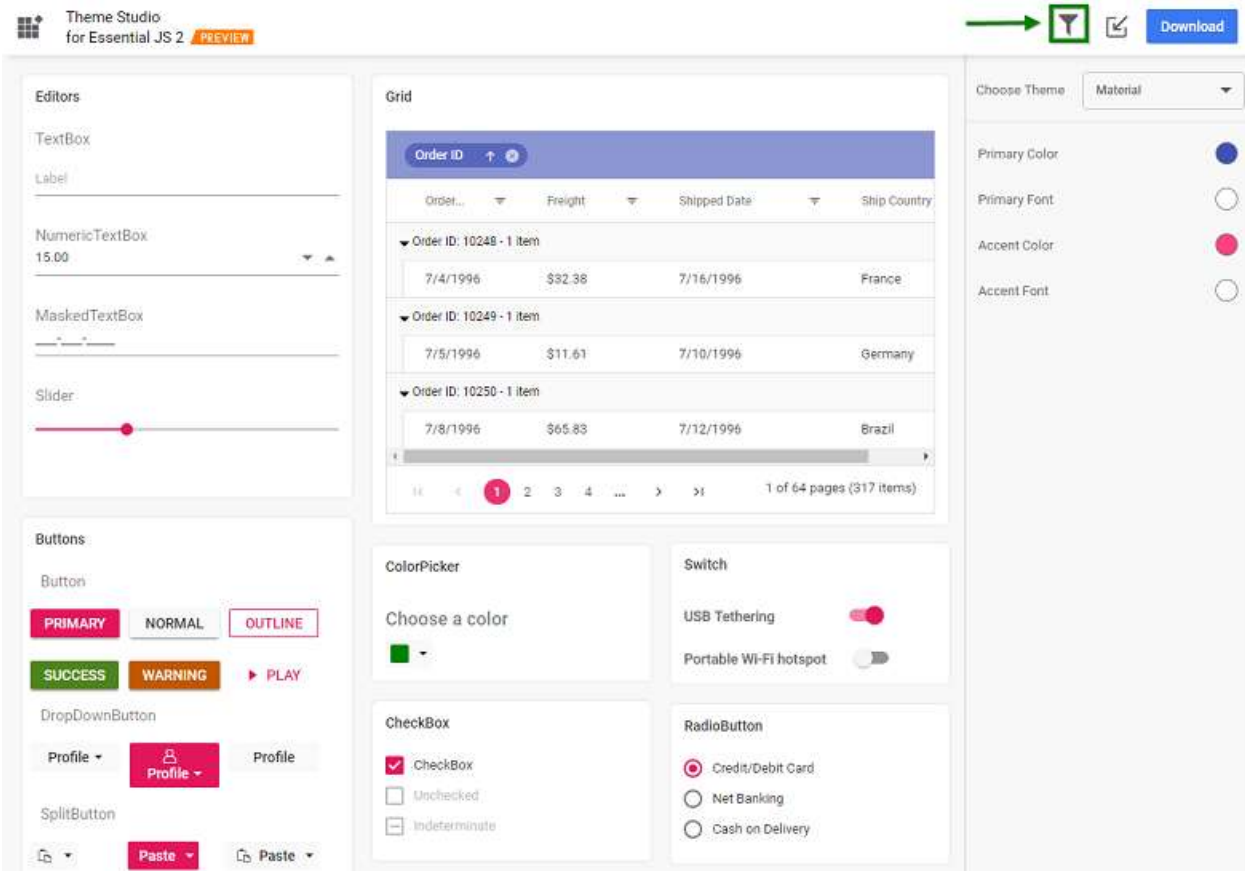
Step 4: The Syncfusion JavaScript UI controls will be rendered with the newly selected colors in the preview section, after selecting the custom color from pickers.



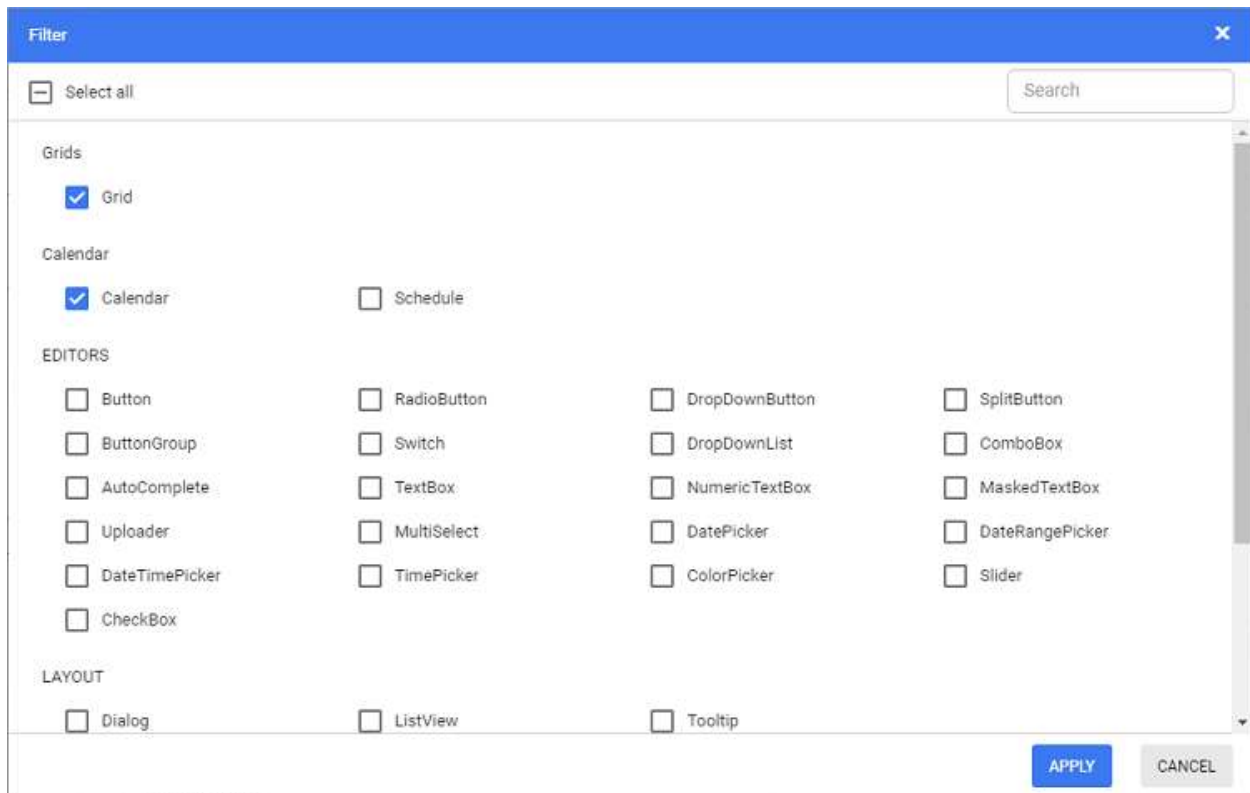
Filtering a specific list of controls

Using the theme studio, you can apply custom themes to a list of specific controls. This option is useful when you have integrated a selective list of Syncfusion JavaScript UI controls in your application. The theme studio will filter the selected controls and customize the final output for those controls' styles alone, reducing the final output file size.

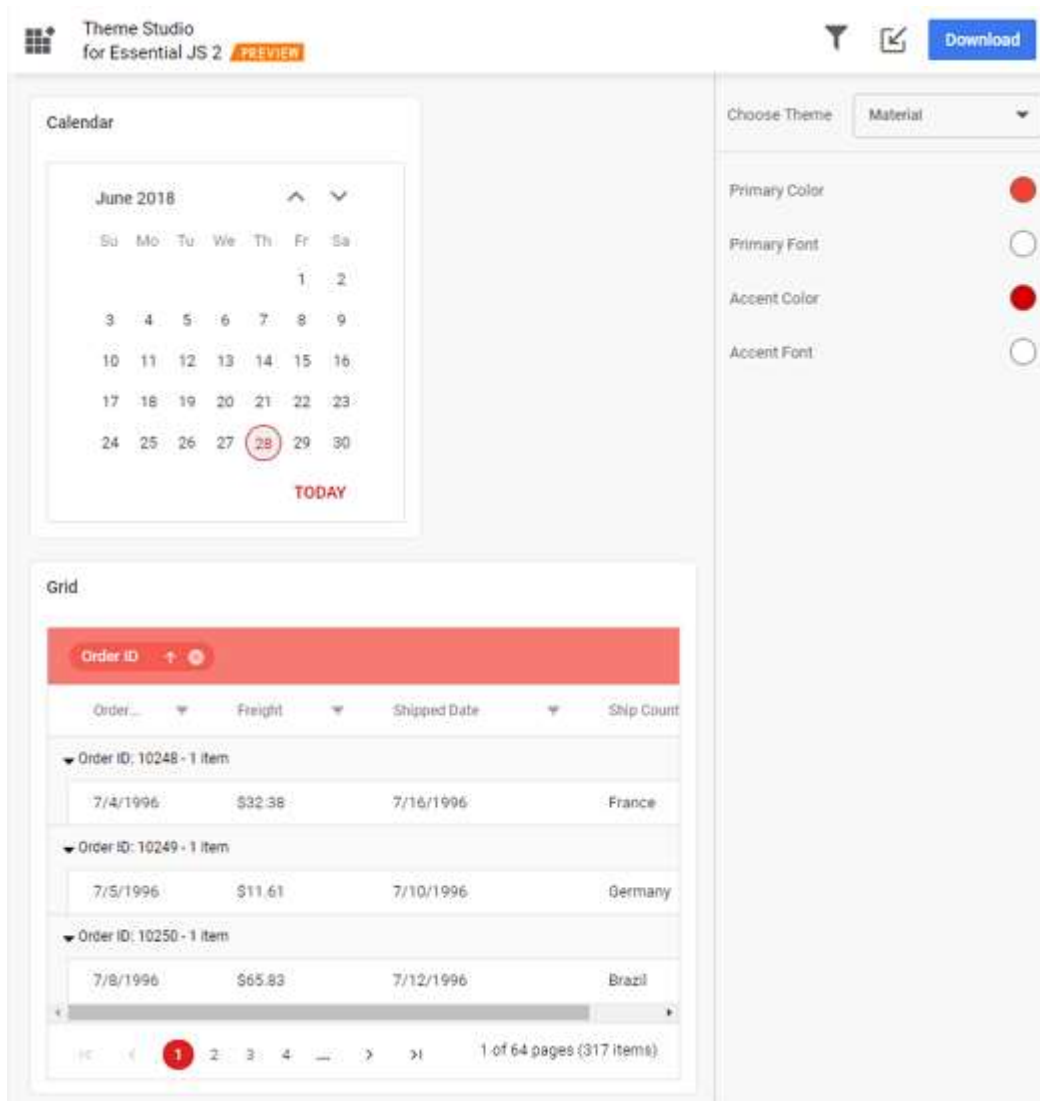
Step 1: Click the Filter icon in the top right corner and select the controls whose theme you want to customize.



Step 2: Click the Apply button in the Filter dialog. Now, only the selected controls will be rendered in the controls preview section.



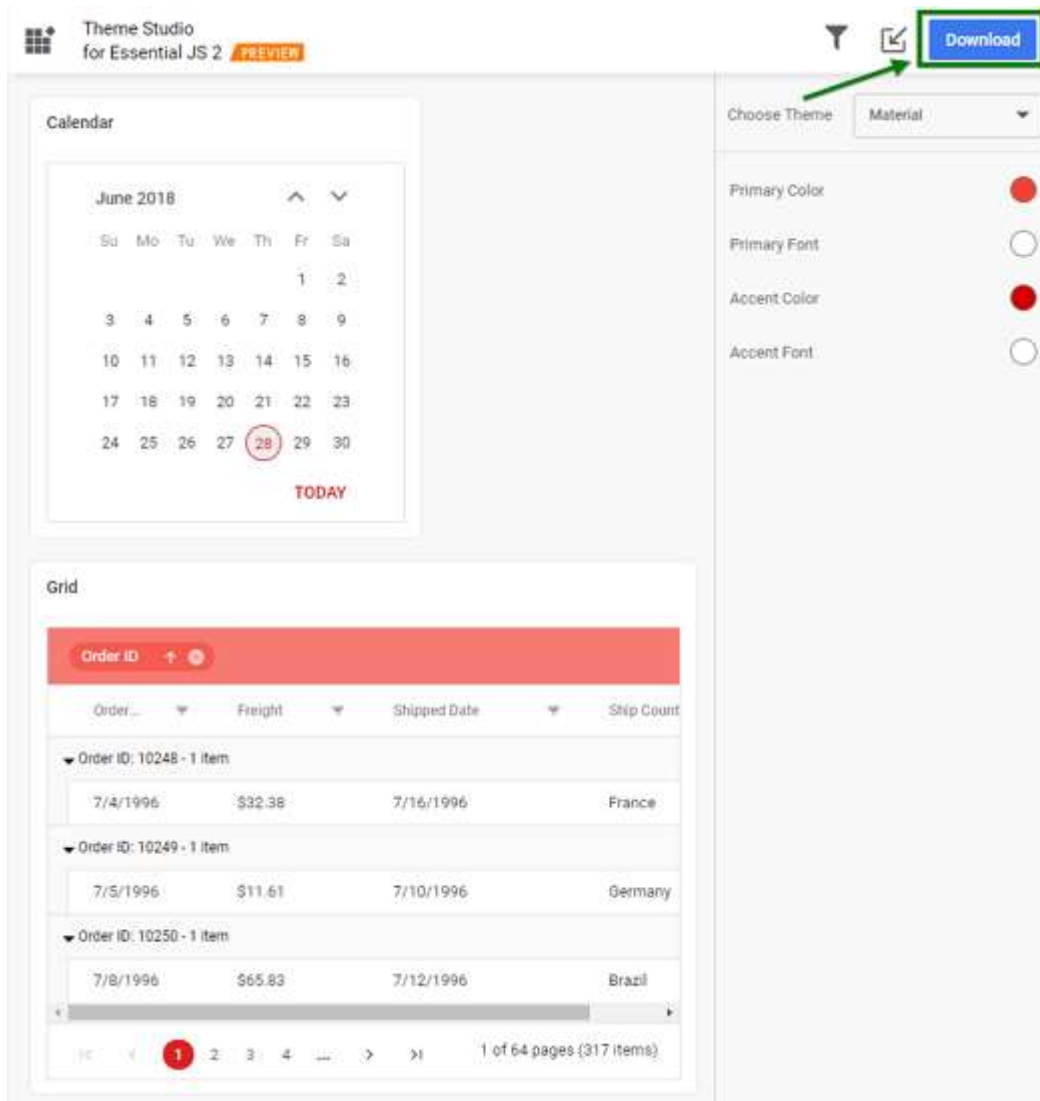
Step 3: Now you can customize the colors in the theme customization section for the controls you selected.



[Download the customized theme](#)

You can download the custom styles after customizing the theme colors.

Step 1: Click the Download button in the top right corner. The Download dialog will open.



Step 2: Assign a theme name in the File Name field and click the Download button. If your application uses both Essential JS 1 and Essential JS 2 controls, then select the Include compatibility css check box before downloading the theme. This option will generate the custom theme for Essential JS 2 compatibility styles, which are compatible as Essential JS 1 styles. Refer this [link](#) for more details about Essential JS 1 and Essential JS 2 compatibility.

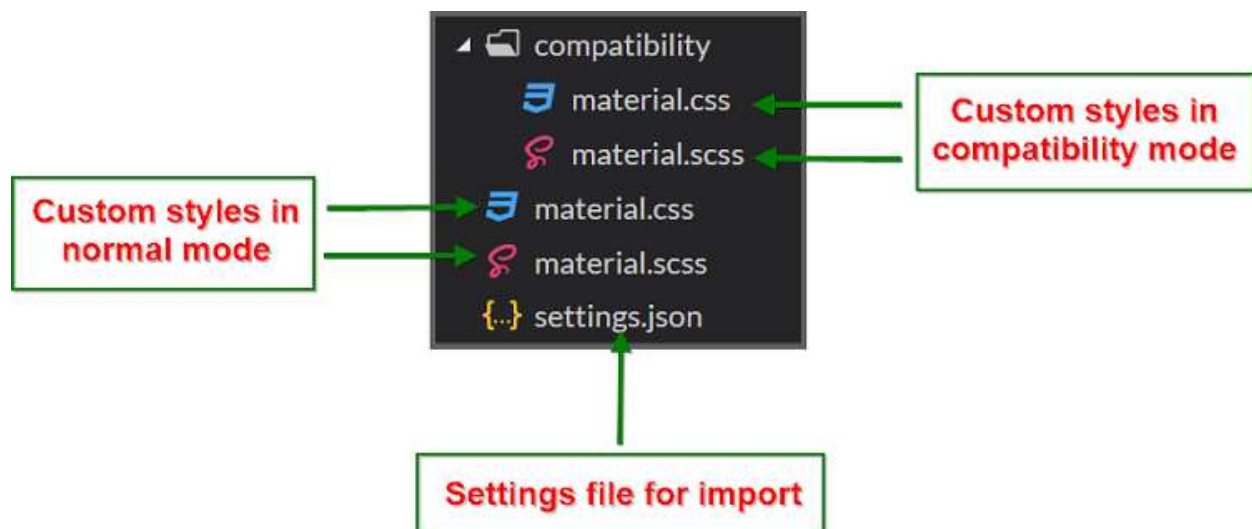
Download [X]

File Name
material

☐ Include compatibility css ⓘ

DOWNLOAD CANCEL

Step 3: The download styles will come as a zip file that contains SCSS and CSS files for the selected Syncfusion JavaScript UI controls. The current settings are stored in the `settings.json` file.



Using customized theme in a web application

You can directly use the customized CSS file in the web application.

Step 1: Copy/paste the customized CSS file from the download folder into your application at any folder.
Example: `styles\{file-name}.css`.

Step 2: Refer the customized CSS file reference in the `index.html` or `shared/_layout.cshtml` main page head section.

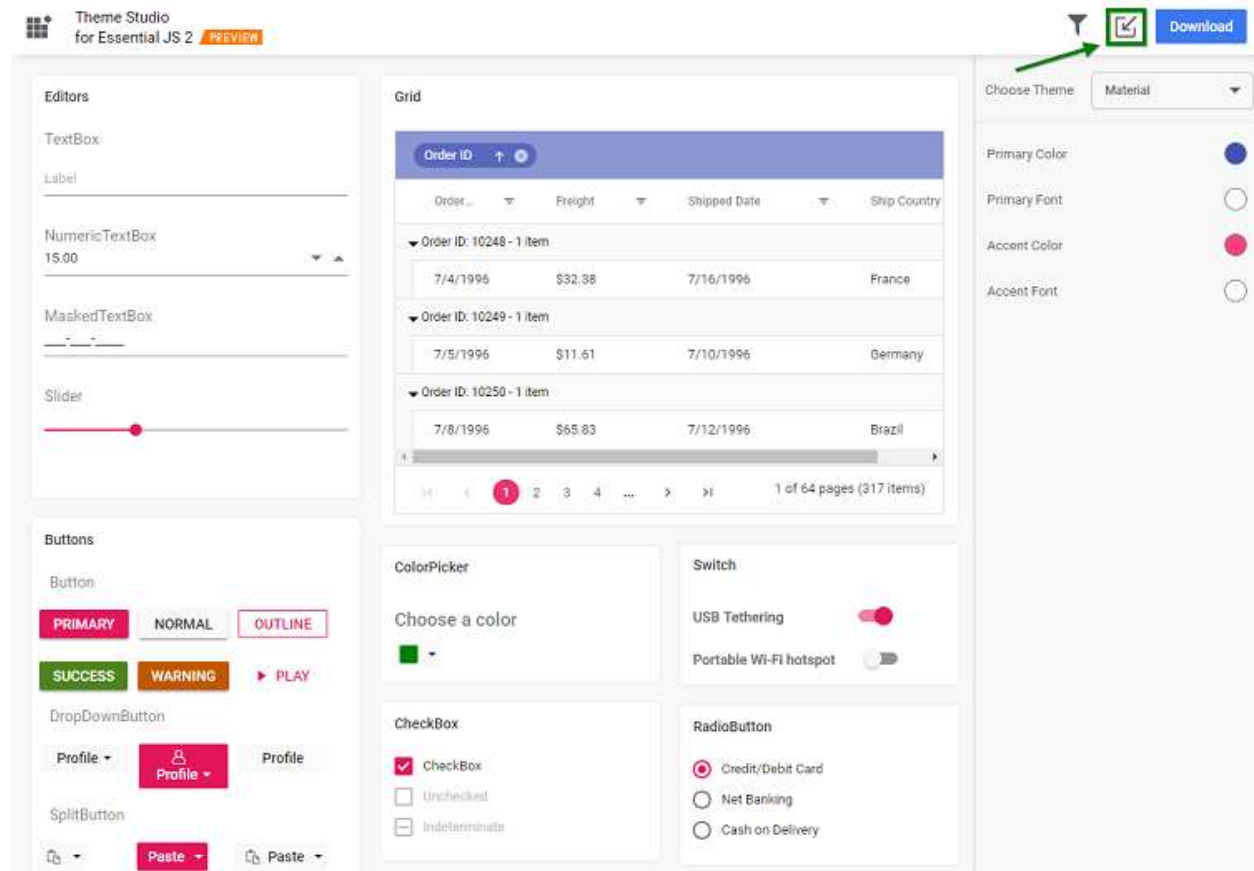
```
<head>
<link href="styles/{file-name}.css" rel="stylesheet"/>
</head>
```

If you are using Essential JS 1 and Essential JS 2 controls in a same web application, then you have to copy/paste the customized CSS file from the `compatibility` folder in the download location.

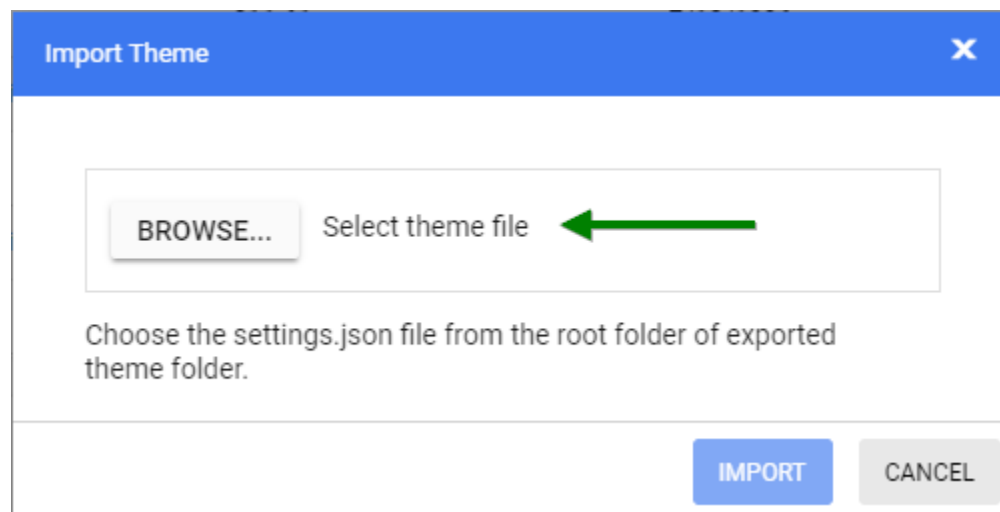
Import previously changed settings into the theme studio

When you want to change your application theme and UI design in the future, you won't need to customize the Syncfusion JavaScript UI controls from scratch in the theme studio. Just import the old `settings.json` file to review and update your stored settings in the theme studio application.

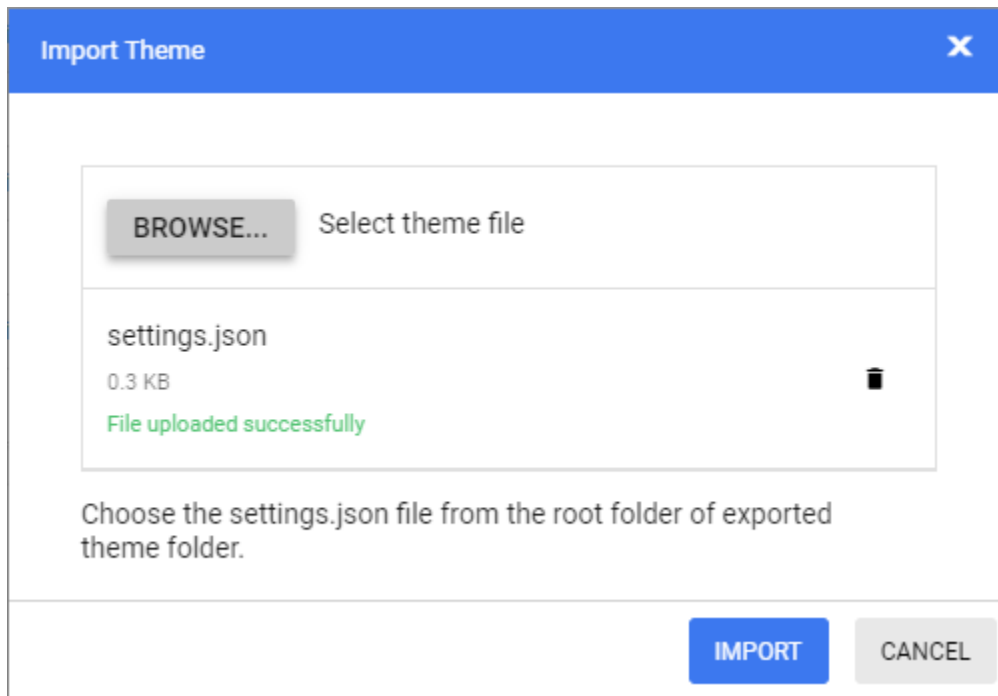
Step 1: Click the Import icon in the top right corner.



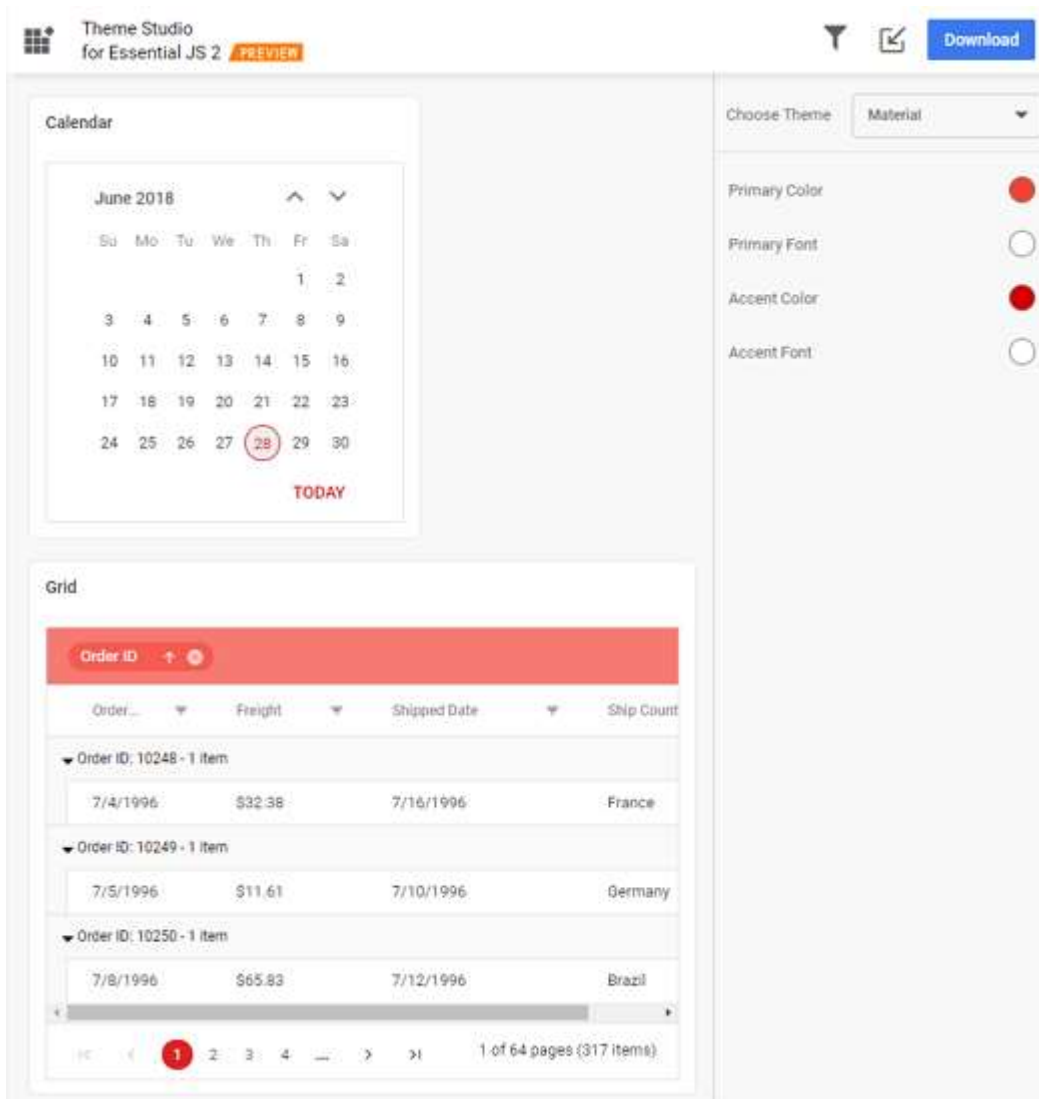
Step 2: The Import Theme dialog will open. Click the Browse button and select a `settings.json` file you exported previously.



Step 3: Click the Import button.



Step 4: The stored data will be reflected in the theme studio application. Now you can change the theme colors based on your latest design and export the theme again.



Step 5: The exported file will contain your latest changes. You can just replace the older custom style with the newer one to refresh your application.

Material 3 Theme

Material 3 is an updated theme that encompasses enhanced theming, components, and personalization features, including dynamic color capabilities. It has been specifically designed to align seamlessly with the new visual style and system UI introduced in Android 12 and above. For more detailed information, please refer to the following link: [Link to Material 3 Information](#).

Syncfusion Material 3 Theme

Syncfusion has introduced the Material theme across all EJ2 Controls. The theme includes light and dark variants and utilizes CSS variables for easy customization of control colors in CSS format. This implementation enables seamless switching between light and dark color schemes, offering users a flexible solution to suit their preferences and application needs.

Note: Please be aware that in the Material 3 theme, we utilize CSS variables with `rgb()` values for our color variables. Using hex values in this context may lead to improper functionality. For example, in previous versions of our Material theme or other themes, the primary color variable was defined as

follows: `$primary: #6200ee;`. In the Material 3 theme, the primary color variable is defined as follows: `--color-sf-primary: 98, 0, 238;`.

What are CSS Variables?

CSS variables, also known as custom properties, are a powerful feature in CSS that allows you to define reusable values and apply them throughout your stylesheets. Prefixed with "--", CSS variables can be utilized in any property value within a CSS rule. To retrieve the value of a CSS variable, the `var()` function is used. CSS variables can access the Document Object Model (DOM), enabling the creation of variables with either local or global scope. For further details, please consult the following link: [Link to CSS Variables Information](#).

How does Syncfusion Theming utilize CSS Variables?

The Material 3 theme in our system incorporates CSS variable support, utilizing CSS variables with `rgb()` values for color customization.

```
`CSS
:root {
--color-sf-black: 0, 0, 0;
--color-sf-white: 255, 255, 255;
--color-sf-primary: 103, 80, 164;
--color-sf-primary-container: 234, 221, 255;
--color-sf-secondary: 98, 91, 113;
--color-sf-secondary-container: 232, 222, 248;
--color-sf-tertiary: 125, 82, 96;
--color-sf-tertiary-container: 255, 216, 228;
--color-sf-surface: 255, 255, 255;
--color-sf-surface-variant: 231, 224, 236;
--color-sf-background: var(--color-sf-surface);
--color-sf-on-primary: 255, 255, 255;
--color-sf-on-primary-container: 33, 0, 94;
--color-sf-on-secondary: 255, 255, 255;
--color-sf-on-secondary-container: 30, 25, 43;
--color-sf-on-tertiary: 255, 255, 255;
}
```

Exploring Color Customization

Through the utilization of these CSS variables, customization of the color variables becomes remarkably straightforward. For example, to customize the primary color variable in this theme, simply modify its value in the root values.

Default primary value

```
--color-sf-black: 0, 0, 0;
--color-sf-white: 255, 255, 255;
--color-sf-primary: 103, 80, 164;
--color-sf-primary-container: 234, 221, 255;
--color-sf-secondary: 98, 91, 113;
--color-sf-secondary-container: 232, 222, 248;
--color-sf-tertiary: 125, 82, 96;
--color-sf-tertiary-container: 255, 216, 228;
--color-sf-surface: 255, 255, 255;
--color-sf-surface-variant: 231, 224, 236;
```

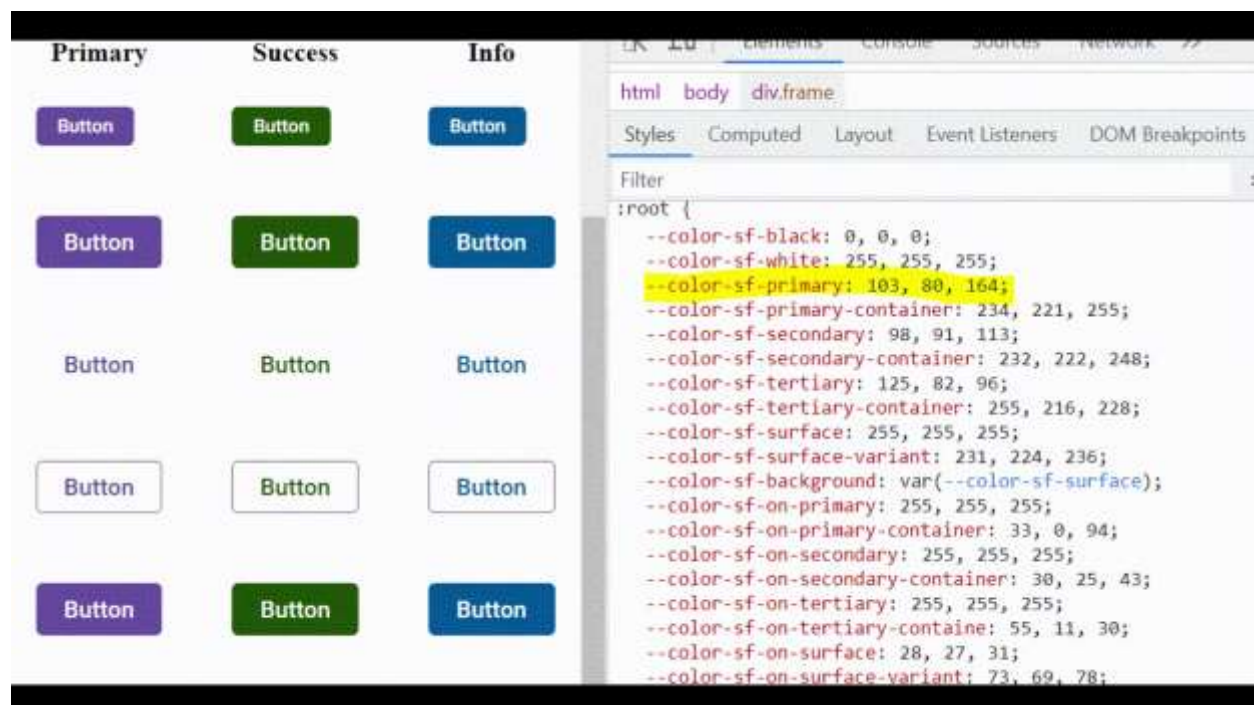
Customized primary value

```
--color-sf-black: 0, 0, 0;
--color-sf-white: 255, 255, 255;
--color-sf-primary: 236, 80, 164;
--color-sf-primary-container: 234, 221, 255;
--color-sf-secondary: 98, 91, 113;
--color-sf-secondary-container: 232, 222, 248;
--color-sf-tertiary: 125, 82, 96;
--color-sf-tertiary-container: 255, 216, 228;
--color-sf-surface: 255, 255, 255;
--color-sf-surface-variant: 231, 224, 236;
```

With this CSS variable support, you can effortlessly customize the color variable values for our EJ2 controls.

Dark mode support

The controls in our system now seamlessly transition between light and dark modes without any noticeable delay. This achievement was made by consolidating the configurations of the light theme and dark theme into [material 3 light theme](#) file.



INDEX.JS

```

var checkBoxObj = new ej.buttons.CheckBox({ label: 'Enable DarkMode',
change: onDarkMode });
checkBoxObj.appendTo('#darkmode');
function onDarkMode(e) {
    e.checked ? document.body.classList.add('e-dark-mode') :
document.body.classList.remove('e-dark-mode');
}

var btnObj = new ej.buttons.Button({ isPrimary: true });
btnObj.appendTo('#normal4');
btnObj = new ej.buttons.Button({ cssClass: 'e-success' });
btnObj.appendTo('#normal5');
btnObj = new ej.buttons.Button({ cssClass: 'e-info' });
btnObj.appendTo('#normal6');
btnObj = new ej.buttons.Button({ cssClass: 'e-warning' });
btnObj.appendTo('#normal7');
btnObj = new ej.buttons.Button({ cssClass: 'e-danger' });
btnObj.appendTo('#normal8');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Material3</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="./index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/material3.css"
rel="stylesheet">
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div style="padding: 10px; display: inline-block;">
            <input id="darkmode" type="checkbox" />
        </div>
        <div class="frame">
            <table id="basic-button">
                <tbody>
                    <tr>
                        <td><button id="normal4">Button</button></td>
                        <td><button id="normal5">Button</button></td>
                        <td><button id="normal6">Button</button></td>
                        <td><button id="normal7">Button</button></td>
                        <td><button id="normal8">Button</button></td>
                    </tr>
                </tbody>
            </table>
        </div>
    </div>
<script>
var ele = document.getElementById('container');

```

```

if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="./index.js" type="text/javascript"></script>
</body></html>

```

The above example demonstrates the appearance of an application in dark mode. By using CSS variable support, the transition between light and dark mode is smooth and visually pleasing.

`CSS

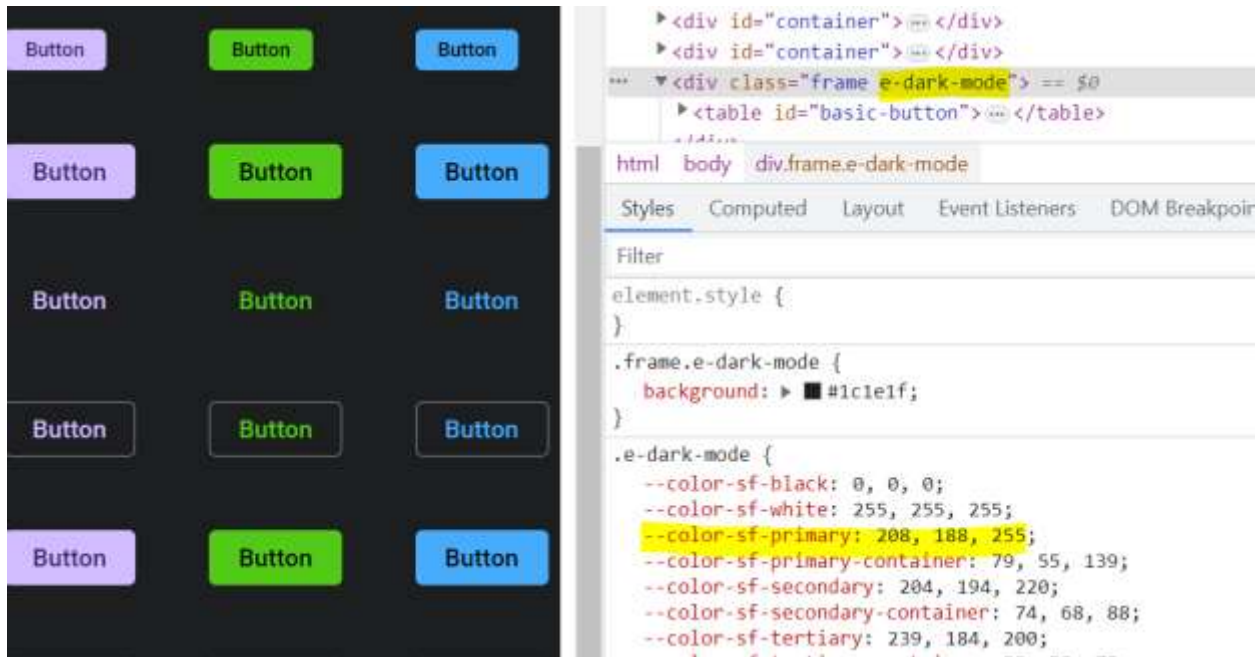
```

.e-dark-mode {
--color-sf-black: 0, 0, 0;
--color-sf-white: 255, 255, 255;
--color-sf-primary: 208, 188, 255;
--color-sf-primary-container: 79, 55, 139;
--color-sf-secondary: 204, 194, 220;
--color-sf-secondary-container: 74, 68, 88;
--color-sf-tertiary: 239, 184, 200;
--color-sf-tertiary-container: 99, 59, 72;
--color-sf-surface: 28, 27, 31;
--color-sf-surface-variant: 28, 27, 31;
--color-sf-background: var(--color-sf-surface);
--color-sf-on-primary: 55, 30, 115;
--color-sf-on-primary-container: 234, 221, 255;
--color-sf-on-secondary: 51, 45, 65;
--color-sf-on-secondary-container: 232, 222, 248;
--color-sf-on-tertiary: 73, 37, 50;
}
`

```

[How to switch to dark mode](#)

With this CSS variable support, transitioning between light and dark theme modes has become effortless. To switch to dark mode, simply add the `e-dark-mode` class to the body section of your application. Upon adding this class, the theme will seamlessly switch to dark mode. Please refer to the example image below for guidance.



ThemeStudio application

The ThemeStudio application now includes seamless integration with the Material 3 theme, offering a comprehensive solution for customization requirements. This enhancement enables users to effortlessly customize and personalize their themes.

Access the Syncfusion ThemeStudio application, featuring the Material 3 theme, via the following link:
[Link to Syncfusion ThemeStudio with Material 3 Theme](#)

Common

Accessibility in Syncfusion ##Platform_Name## controls

Accessibility overview

Accessibility in controls refers to the practice of designing and building user interface elements in a way that makes them accessible to users with disabilities. This can include a variety of things, such as making sure that text is high-contrast and easy to read, providing alternative text for images, and designing controls and interactions that can be used with a keyboard or assistive technology.

Accessibility standards

The control is said to be accessible when it is constructed in accordance with certain standards that are required to make it accessible.

The accessibility of the controls consists of the following standards and aspects:

- [ADA](#) - A law to ensure that people with disabilities have the same opportunities and access as people without disabilities.
- [WCAG 2.2](#) - The Web Content Accessibility Guidelines (WCAG) provide guidelines developed by the World Wide Web Consortium (W3C) to ensure web content is accessible to people with disabilities. WCAG 2.2 establishes a framework of accessibility principles and their associated success criteria. The level of accessibility conformance achieved by a web application is determined by the extent to which it meets these success criteria, categorized into three levels: A, AA, and AAA.

- [Section 508](#) - It is a set of guidelines for making electronic and information technology (EIT) accessible to people with disabilities. These standards apply to federal agencies in the United States, and they are based on the Web Content Accessibility Guidelines (WCAG).
- [WAI-ARIA](#) - WAI-ARIA stands for "Web Accessibility Initiative - Accessible Rich Internet Applications." It is a set of technical specifications and guidelines developed by the World Wide Web Consortium (W3C) as part of the Web Accessibility Initiative (WAI). WAI-ARIA is designed to enhance the accessibility of dynamic web content, particularly web applications and rich internet applications (RIAs), for people with disabilities. WAI-ARIA provides a set of roles, states, and properties that can be added to HTML elements to provide additional context and information about the purpose and behavior of those elements. This can help assistive technologies better understand and interpret web content and interact with web applications.
- [Keyboard navigation](#) - It refers to the ability to use a keyboard to interact with and navigate through a user interface. It is an important aspect of web accessibility, as it allows people who cannot use a mouse or other pointing device to access and use web content and applications.

Syncfusion ##Platform_Name## controls adhere to these established standards.

Accessibility compliance

The accessibility support provided by Syncfusion ##Platform_Name## controls is based on a collection of methodologies for adhering to and applying [recognized standards and technical specifications](#) to ensure an intuitive experience for people with disabilities.

There are several methodologies of accessibility validation that can be performed on the Syncfusion ##Platform_Name## controls, such as:

- The [WAI-ARIA patterns](#) are followed by the Syncfusion ##Platform_Name## controls to enable appreciable accessibility.
- Each ##Platform_Name## control is subjected to manual testing with a screen reader and also automated test cases to ensure the control's required attributes.
- Attributes are allocated and updated correctly during interaction as well. Each control has been assigned a distinct **Role** attribute and its own set of ARIA attributes defined by the [WCAG 2.2](#) specification.

In addition to the methodologies mentioned above, Syncfusion ##Platform_Name## controls are constructed to support the following accessibility aspects.

Screen reader support

A screen reader allows people who are blind or visually impaired to use a computer by reading aloud the text that is displayed on the screen. Syncfusion ##Platform_Name## controls followed the [WAI-ARIA](#) standards to work properly in the screen readers such as [Narrator](#) for Windows and [Embedded VoiceOver](#) for MAC.

Right-To-Left support

Right-to-Left (RTL) support allows applications to effectively communicate with users who use languages that are written from right to left, such as Arabic, Hebrew, etc. Syncfusion ##Platform_Name## controls support the Right-to-Left feature. Refer to the [Right-to-Left documentation](#) to learn more about this support.

Color contrast

Syncfusion `##Platform_Name##` controls come equipped with [predefined themes](#) that guarantee the presence of satisfactory color contrast, benefiting individuals with visual impairments.

Mobile device support

Syncfusion `##Platform_Name##` controls are more user-friendly and accessible to individuals using mobile devices, including those with disabilities. These are designed to be responsive, adaptable to various screen sizes and orientations, and touch-friendly.

Keyboard navigation support

Syncfusion `##Platform_Name##` controls support keyboard navigation, allowing users who rely on alternate methods to effortlessly navigate and interact with the control.

Ensuring accessibility

Ensuring the accessibility of Syncfusion `##Platform_Name##` controls is crucial for making the product inclusive and user-friendly for individuals with disabilities. This process involves various types of accessibility testing, including:

- **Automated testing:** The Syncfusion `##Platform_Name##` control's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools.
- **Manual testing:** This type of testing involves manually evaluating the Syncfusion `##Platform_Name##` controls. During manual accessibility testing, testers will ensure accessibility using the screen readers such as [Narrator](#) for Windows and [Embedded VoiceOver](#) for MAC.

Syncfusion `##Platform_Name##` controls will keep improving when there is anything required. It also involves client feedback to make the control more accessible.

Accessibility support for specific controls

Consult the control-specific documents below for detailed information about how Syncfusion `##Platform_Name##` controls ensure compliance with accessibility standards, encompassing Section 508, WAI-ARIA, WCAG 2.2, keyboard navigation, and more.

<style>

table

{

border:0 !important;

line-height: 2!important;

}

tr

{

border:0 !important;

}

td

{

```

border:0 !important;
vertical-align: top;
}
.controlanchorlink
{
text-decoration: none !important;
font-size: 14px !important;
text-align: left !important;
padding: 5px 0px;
letter-spacing: 1px;
}
.controlcategory
{
font-size: 14px !important;
text-align: left !important;
font-weight: bold !important;
letter-spacing: 0.7px;
}
}
}
</style>

```

| GRIDS | DATA
VISUALIZATION | CALENDARS | DROPDOWNS |
|--------------------------------|------------------------------------|---------------------------------|--------------------------------------|
| DataGrid | Accumulation Chart | Scheduler | AutoComplete |
| Pivot Table | Charts | Gantt Chart | ListBox |
| TreeGrid | Stock Chart | Calendar | ComboBox |
| Spreadsheet | Circular Gauge | DatePicker | DropDown List |
| FILE VIEWERS &
EDITORS | Linear Gauge | DateRangePicker | Multiselect DropDown |
| RichTextEditor | Maps | DateTime Picker | DropDown Tree |
| PDF Viewer | Diagram | TimePicker | Mention |
| LAYOUT | Range Selector | INPUTS | NAVIGATION |
| Dialog | Smith Chart | Input Mask | Accordion |
| | Sparkline Charts | Numeric TextBox | Carousel |
| | TreeMap | | |

| | | | |
|---------------------------|---------------------------------|--------------------------------------|------------------------------|
| ListView | Bullet Chart | Masked TextBox | Context Menu |
| Tooltip | BUTTONS | RadioButton | Menu Bar |
| Splitter | ButtonGroup | CheckBox | Tabs |
| Dashboard | Dropdown Menu | Color Picker | Toolbar |
| | Progress Button | File Upload | TreeView |
| | SplitButton | Range Slider | File Manager |
| | Chips | Toggle Switch Button | Breadcrumb |
| | Speed Dial | Signature | Pager |
| | | Rating | NOTIFICATION |
| | | | Toast |
| | | | Skeleton |
| | | | Message |

Animation in ##Platform_Name##

The **Animation** is used to perform animation effects on HTML elements by running a sequence of frames. It can be used to enhance the user experience.

Syncfusion [Animation](#) library supports animating the HTML elements using the [animate](#) method. This method adds the `e-animate`, `e-animation-id` attributes, and CSS style to the HTML element and removes them after the animation effect is completed.

Animation effects

An animation effect refers to the visual change that occurs over a period of time in HTML elements. The [Animation](#) library supports different types of animation [effects](#). The [name](#) property is used to specify the animation effect of an animation.

Here is an example code snippet using the `FadeOut` and `ZoomOut` animation effects:

INDEX.JS

```
var animation = new ej.base.Animation();
animation.animate('#element1', { name: 'FadeOut' });
animation.animate('#element2', { name: 'ZoomOut' });
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element1"></div>
        <div id="element2"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Animation duration

Animation [duration](#) is the animation property that specifies the length of time that an animation should take to complete. The animation duration is specified in milliseconds (ms) and determines the total amount of time that an animation should run.

For example, if an animation has a duration of 2 seconds, it will take 2 seconds to complete from start to finish. The duration of an animation affects the overall pace of the animation and can be adjusted to match the desired speed and style of the animation.

The value of the animation duration can be adjusted to change the speed of the animation, with shorter durations resulting in faster animations and longer durations resulting in slower animations.

Here is an example code snippet using the animation effects with a duration of **5000** milliseconds:

INDEX.JS

```

var animation = new ej.base.Animation({ duration: 5000 });
animation.animate('#element1', { name: 'FadeOut' });
animation.animate('#element2', { name: 'ZoomOut' });

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element1"></div>
        <div id="element2"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Animation delay

The animation [delay](#) is the animation property that specifies the amount of time to wait before starting an animation. The animation delay is specified in milliseconds (ms) and determines the amount of time that should elapse before an animation begins.

For example, if an animation has a delay of 2 seconds, it will wait for 2 seconds before starting. This can be useful in creating more complex animations, where multiple elements are animated in sequence, or in creating animations that start only after a user interaction has taken place.

Here is an example code snippet using the animation effects with a delay of **2000** milliseconds:

INDEX.JS

```

var animation = new ej.base.Animation({ delay: 2000, duration: 5000 });
animation.animate('#element1', { name: 'FadeOut' });
animation.animate('#element2', { name: 'ZoomOut' });

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>

```

```
<body>

  <div id="container">
    <div id="element1"></div>
    <div id="element2"></div>
  </div>
</script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Enable or disable animation globally

Enable or disable animation for all ##Platform_Name## controls globally by using the `setGlobalAnimation` method with one of the below options:

- `GlobalAnimationMode.Enable` - Enables the animation for all components, regardless of the individual component's animation settings.
- `GlobalAnimationMode.Disable` - Disables the animation for all components, regardless of the individual component's animation settings.
- `GlobalAnimationMode.Default` - Animation is enabled or disabled based on the component's animation settings.

In the below code snippet, animation is disabled.

INDEX.JS

```
ej.base.setGlobalAnimation(ej.base.GlobalAnimationMode.Disable);
```

`setGlobalAnimation` method controls script-level animations only, and it is not applicable for direct CSS-level animations (animations defined from CSS classes or properties).

Right-To-Left support in Syncfusion JavaScript Controls

Right-to-Left (RTL) support allows applications to effectively communicate with users who use languages that are written from right to left, such as Arabic, Hebrew, etc.

Syncfusion JavaScript (ES5) controls support for right-to-left (RTL) by setting the `enableRtl` property to `true`. This adds the class name `e-rtl` to the control element and renders all Syncfusion JavaScript controls in a right-to-left direction.

Enable RTL for all controls

To enable Right-To-Left (RTL) support for all controls, users can set the `enableRtl` property directly in their application. Here is an example code snippet using the `ListView` control:

INDEX.JS

```
// Enables Right to left alignment for all controls
ej.base.enableRtl(true);
var rtlListObj = new ej.lists.ListView({
```

```

        dataSource: data,
        headerTitle: 'کاری',
        showHeader: true,
        height: '350px'
    });
    rtlListObj.appendTo('#listview');

```

INDEX.HTML

```

<!DOCTYPE html><html><head>
    <meta charset="UTF-8">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="listview"></div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

Enable RTL for an individual control

To enable Right-To-Left (RTL) support for an individual control, users can set the [enableRtl](#) property in the control's options. Here is an example code snippet using the ListView control:

INDEX.JS

```

var rtlListObj = new ej.lists.ListView({
    dataSource: data,
    enableRtl: true,
    headerTitle: 'کاری',
    showHeader: true,
    height: '350px'
});
rtlListObj.appendTo('#listview');

```

INDEX.HTML

```

<!DOCTYPE html><html><head>
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="listview"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

State Persistence in Syncfusion JavaScript controls

Syncfusion JavaScript controls support persisting their state across page refreshes or navigation. To enable this feature, set the [enablePersistence](#) property to true for the desired control. This stores the control's state in the browser's `localStorage` object on the `unload` event of the page. For example, the [enablePersistence](#) property can be set for the Grid control, as shown in the following code snippet.

INDEX.JS

```

var data = [
    {
        OrderID: 10248, CustomerID: 'VINET', Role: 'Admin', EmployeeID: 5,
        OrderDate: new Date(8364186e5),
        ShipName: 'Vins et alcools Chevalier', ShipCity: 'Reims',
        ShipAddress: '59 rue de l Abbaye',
        ShipRegion: 'CJ', Mask: '1111', ShipPostalCode: '51100', ShipCountry:
        'France', Freight: 32.38, Verified: !0
    },
    {
        OrderID: 10249, CustomerID: 'TOMSP', Role: 'Employee', EmployeeID:
        6, OrderDate: new Date(836505e6),
        ShipName: 'Toms Spezialitäten', ShipCity: 'Münster', ShipAddress:
        'Luisenstr. 48',
        ShipRegion: 'CJ', Mask: '2222', ShipPostalCode: '44087',
        ShipCountry: 'Germany', Freight: 11.61, Verified: !1
    },
    {
        OrderID: 10250, CustomerID: 'HANAR', Role: 'Admin', EmployeeID: 4,
        OrderDate: new Date(8367642e5),
        ShipName: 'Hanari Carnes', ShipCity: 'Rio de Janeiro', ShipAddress:
        'Rua do Paço, 67',
        ShipRegion: 'RJ', Mask: '3333', ShipPostalCode: '05454-876',
        ShipCountry: 'Brazil', Freight: 65.83, Verified: !0
    },

```



```

{
    OrderID: 10251, CustomerID: 'VICTE', Role: 'Manager', EmployeeID: 3,
    OrderDate: new Date(8367642e5),
    ShipName: 'Victuailles en stock', ShipCity: 'Lyon', ShipAddress: '2,
    rue du Commerce',
    ShipRegion: 'CJ', Mask: '4444', ShipPostalCode: '69004',
    ShipCountry: 'France', Freight: 41.34, Verified: !0
},
{
    OrderID: 10252, CustomerID: 'SUPRD', Role: 'Manager', EmployeeID: 2,
    OrderDate: new Date(8368506e5),
    ShipName: 'Suprêmes délices', ShipCity: 'Charleroi', ShipAddress:
    'Boulevard Tirou, 255',
    ShipRegion: 'CJ', Mask: '5555', ShipPostalCode: 'B-6000',
    ShipCountry: 'Belgium', Freight: 51.3, Verified: !0
},
{
    OrderID: 10253, CustomerID: 'HANAR', Role: 'Admin', EmployeeID: 7,
    OrderDate: new Date(836937e6),
    ShipName: 'Hanari Carnes', ShipCity: 'Rio de Janeiro', ShipAddress:
    'Rua do Paço, 67',
    ShipRegion: 'RJ', Mask: '6666', ShipPostalCode: '05454-876',
    ShipCountry: 'Brazil', Freight: 58.17, Verified: !0
},
{
    OrderID: 10254, CustomerID: 'CHOPS', Role: 'Employee', EmployeeID:
    5, OrderDate: new Date(8370234e5),
    ShipName: 'Chop-suey Chinese', ShipCity: 'Bern', ShipAddress:
    'Hauptstr. 31',
    ShipRegion: 'CJ', Mask: '7777', ShipPostalCode: '3012', ShipCountry:
    'Switzerland', Freight: 22.98, Verified: !1
},
{
    OrderID: 10255, CustomerID: 'RICSU', Role: 'Admin', EmployeeID: 9,
    OrderDate: new Date(8371098e5),
    ShipName: 'Richter Supermarkt', ShipCity: 'Genève', ShipAddress:
    'Starenweg 5',
    ShipRegion: 'CJ', Mask: '8888', ShipPostalCode: '1204', ShipCountry:
    'Switzerland', Freight: 148.33, Verified: !0
},
{
    OrderID: 10256, CustomerID: 'WELLI', Role: 'Employee', EmployeeID:
    3, OrderDate: new Date(837369e6),
    ShipName: 'Wellington Importadora', ShipCity: 'Resende',
    ShipAddress: 'Rua do Mercado, 12',
    ShipRegion: 'SP', Mask: '9999', ShipPostalCode: '08737-363',
    ShipCountry: 'Brazil', Freight: 13.97, Verified: !1
},
{
    OrderID: 10257, CustomerID: 'HILAA', Role: 'Admin', EmployeeID: 4,
    OrderDate: new Date(8374554e5),
    ShipName: 'HILARION-Abastos', ShipCity: 'San Cristóbal',
    ShipAddress: 'Carrera 22 con Ave. Carlos Soublette #8-35',
    ShipRegion: 'Táchira', Mask: '1234', ShipPostalCode: '5022',
    ShipCountry: 'Venezuela', Freight: 81.91, Verified: !0
},
{

```

```

        OrderID: 10258, CustomerID: 'ERNSH', Role: 'Manager', EmployeeID: 1,
        OrderDate: new Date(8375418e5),
        ShipName: 'Ernst Handel', ShipCity: 'Graz', ShipAddress: 'Kirchgasse
6',
        ShipRegion: 'CJ', Mask: '2345', ShipPostalCode: '8010',
        ShipCountry: 'Austria', Freight: 140.51, Verified: !0
    },
    {
        OrderID: 10259, CustomerID: 'CENTC', Role: 'Admin', EmployeeID: 4,
        OrderDate: new Date(8376282e5),
        ShipName: 'Centro comercial Moctezuma', ShipCity: 'México D.F.',
        ShipAddress: 'Sierras de Granada 9993',
        ShipRegion: 'CJ', Mask: '3456', ShipPostalCode: '05022',
        ShipCountry: 'Mexico', Freight: 3.25, Verified: !1
    },
    {
        OrderID: 10260, CustomerID: 'OTTIK', Role: 'Admin', EmployeeID: 4,
        OrderDate: new Date(8377146e5),
        ShipName: 'Ottilies Käseladen', ShipCity: 'Köln', ShipAddress:
'Mehrheimerstr. 369',
        ShipRegion: 'CJ', Mask: '4567', ShipPostalCode: '50739',
        ShipCountry: 'Germany', Freight: 55.09, Verified: !0
    },
    {
        OrderID: 10261, CustomerID: 'QUEDE', Role: 'Manager', EmployeeID: 4,
        OrderDate: new Date(8377146e5),
        ShipName: 'Que Delícia', ShipCity: 'Rio de Janeiro', ShipAddress:
'Rua da Panificadora, 12',
        ShipRegion: 'RJ', Mask: '5678', ShipPostalCode: '02389-673',
        ShipCountry: 'Brazil', Freight: 3.05, Verified: !1
    },
    {
        OrderID: 10262, CustomerID: 'RATTC', Role: 'Employee', EmployeeID:
8, OrderDate: new Date(8379738e5),
        ShipName: 'Rattlesnake Canyon Grocery', ShipCity: 'Albuquerque',
        ShipAddress: '2817 Milton Dr.',
        ShipRegion: 'NM', Mask: '6789', ShipPostalCode: '87110',
        ShipCountry: 'USA', Freight: 48.29, Verified: !0
    }
];
var grid = new ej.grids.Grid({
    dataSource: data,
    enablePersistence: true,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
        { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C' },
        { field: 'OrderDate', headerText: 'Order Date', width: 140, format:
'yMd' }
    ],
    allowPaging: true,
    pageSettings: { pageSize: 7 }
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html><head>

    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="JavaScript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

State Persistence supported controls and properties

The following table illustrates the list of Syncfusion JavaScript controls that are supported with state persistence and the properties that are stored in the `localStorage`.

<!-- markdownlint-disable MD033 -->

control Name	Properties
Grid	<ul style="list-style-type: none"> Columns filterSettings searchSettings groupSettings pageSettings selectedRowIndex scrollPosition
Accordion	<ul style="list-style-type: none"> expandedIndices
Tabs	<ul style="list-style-type: none"> selectedItem
Schedule	<ul style="list-style-type: none"> currentView selectedDate scrollLeft scrollTop
Kanban	<ul style="list-style-type: none"> columns dataSource swimlaneToggleArray
Chart	<ul style="list-style-type: none"> zoomFactor zoomPosition
Maps	<ul style="list-style-type: none"> zoomSettings
Pivot Table	<ul style="list-style-type: none"> dataSourceSettings pivotValues gridSettings chartSettings displayOption
TreeGrid	<ul style="list-style-type: none"> columns pageSettings searchSettings filterSettings selectedRowIndex sortSettings
Switch, Checkbox	<ul style="list-style-type: none"> checked
RadioButton	<ul style="list-style-type: none"> checked

	<ul style="list-style-type: none"> value
ColorPicker, ListBox, In-placeEditor, RichTextEditor, Autocomplete, Calendar, ComboBox, DatePicker, DropDownList, MaskedTextBox, NumericTextBox, Textbox, TimePicker, Multiselect, DateTimePicker, Slider, Dropdown Tree	<ul style="list-style-type: none"> value
QueryBuilder	<ul style="list-style-type: none"> rule
Splitter	<ul style="list-style-type: none"> paneSettings
DateRangePicker	<ul style="list-style-type: none"> startDate endDate value
Uploader	<ul style="list-style-type: none"> filesData
ListView	<ul style="list-style-type: none"> cssClass enableRtl htmlAttributes enable fields animation headerTitle sortOrder showIcon height width showCheckBox checkBoxPosition
TreeView	<ul style="list-style-type: none"> selectedNodes checkedNodes expandedNodes
Dashboard Layout	<ul style="list-style-type: none"> panels
File Manager	<ul style="list-style-type: none"> view path selectedItems
Sidebar	<ul style="list-style-type: none"> type position isOpen

<!-- markdownlint-enable MD033 -->

Check out the following control's document to learn more about the state persistence.

- [Schedule](#)
- [TreeGrid](#)
- [Pivot Table](#)
- [Gantt](#)
- [Grid](#)
- [Kanban](#)
- [QueryBuilder](#)
- [Tabs](#)

Template Engine

Syncfusion JavaScript (Essential JS 2) has built-in template engine which provides options to compile template string into a executable function. Then the generated executable function can be used for rendering DOM element using desired data.

Compiling

`compile` method from `ej2-base` can be used to convert our template strings into executable functions.

INDEX.JS

```
// data
var data = { name: 'Aston Martin' };
// Compiling template string into executable function
var getDOMString = ej.base.compile('<div>${name}</div>');
// Using generated function to get output element collection
var output = getDOMString(data);
// append html collection to element
document.getElementById('element').appendChild(output[0]);
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
```

```

        <div id="element"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

To enable strict Content Security Policy (CSP), it is recommended to utilize the [function template](#) approach instead of the string template.

Available template syntax

Name	Syntax	Description
---	---	---
Expression	<div>\${name}</div>	We have expression evolution as like ES6 expression string literals.
Dot Variable Access	<div>\${person.info.name}</div>	Access the json variable with dot notation.
Variable Function	<div>\${name.toUpperCase()}</div>	Utilize the variable function example, name.toUpperCase()
Window Function	<div>\${getCurrentTime()}</div>	Use window function inside template. Note: Here, getCurrentTime() is a function that defined in the window object.
Custom Helper Function	<div>\${convertToCurrency()}</div>	Use function that passed in helper function.
IF Else Statement	<div> \${if(gender==="male")} Male \${else} Female \${/if} </div>	Branching statement in Template.
For Statement	<div> \${for(mark of marks)} \${mark} \${/for} </div>	Use for looping inside template.
For Index value access	<div> \${for(mark of marks)} \${markIndex} \${/for} </div>	Get the index value of item while using for statement. Use the variable Index that suffixed with loop item name.

Custom helper

Custom helper function can be defined and passed to `compile` function. Refer to the following example.

INDEX.JS

```

var customHelper = {
    upperCase: function upperCase(str) {
        return str.toUpperCase();
    }
};

```

```

var data = { name: 'Aston Martin' };
var getDOMString = ej.base.compile('<div>${upperCase(name)}</div>',
customHelper);
let opElem = getDOMString(data);
document.getElementById('element').appendChild(opElem[0]);

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Template in Syncfusion ##Platform_Name## controls

Syncfusion JavaScript controls are rendered with a pre-defined layout or structure that is used to define how the control should be rendered on the user interface. The user wants to customise the appearance of the control and add functionality that is specific to the needs of the application. Syncfusion JavaScript controls have the option to achieve this using template support. A template can contain a variety of elements, depending on the context in which it is being used.

Types of templates

Syncfusion JavaScript controls have two types of templates, such as:

- [String template](#)
- [Script template](#)
- [Function template](#)

String template

Users can add templates to Syncfusion JavaScript controls by using **string literals** and JavaScript expressions. Using this approach, the template string is passed to the library's template engine, which parses the string and generates the corresponding HTML elements along with the data bindings.

The template string can be added directly to the **template** property of the control. Refer to the following code snippet.

INDEX.JS

```
var grid = new ej.grids.Grid({
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right', width:
125 },
    { field: 'CustomerName', headerText: 'Customer Name', width: 125 },
    { headerText: 'ShipCountry', template: '<div>${ShipCountry}</div>',
width: 125 },
  ],
  dataSource: [
    { OrderID: 10248, ShipCountry: "France", CustomerName: "Paul Henriot" },
    { OrderID: 10249, ShipCountry: "Germany", CustomerName: "Karin Josephs"
},
    { OrderID: 10250, ShipCountry: "Brazil", CustomerName: "Mario Pontes" },
    { OrderID: 10251, ShipCountry: "France", CustomerName: "Mary Saveley" }
  ],
  width: 'auto',
  height: 359
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container">
    <div id="Grid"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
```

```

}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Script template

A script template is a type of template that uses a scripting language, such as JavaScript, to define the structure and layout of the content that is displayed in the control. These template elements can be defined in the `script` tag along with the unique identifier. The script template's identifier needs to be mapped to the corresponding control's template property along with the fragment identifier (#). Refer to the below code sample.

Add the below HTML template to the `index.html` file.

```

`html
<script id="customTemplate" type="text/x-template">
<tr>
<td class="photo">
${EmployeeID}
</td>
<td class="details">

```

First Name	\${FirstName}
Last Name	\${LastName}
Title	\${Title}
Country	\${Country}

```

</td>
</tr>
</script>
`

```

Here, the script template identifier (customTemplate) is assigned to the `template` property of the Grid control. Refer to the following code snippet.

INDEX.JS

```

var grid = new ej.grids.Grid({
  dataSource: employeeData,
  rowTemplate: '#customTemplate',
  columns: [
    { headerText: 'Employee ID', width: 150, textAlign: 'Center', field: 'OrderID' },
    { headerText: 'Employee Details', width: 300, field: 'EmployeeID' }
  ]
});

```

```

    ],
    height: 315
  });
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
  <script src="es5-datasource.js"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <script id="customTemplate" type="text/x-template">
    <tr>
      <td class="photo">
        ${EmployeeID}
      </td>
      <td class="details">
        <table class="CardTable" cellpadding="3" cellspacing="2">
          <colgroup>
            <col width="50%">
            <col width="50%">
          </colgroup>
          <tbody>
            <tr>
              <td class="CardHeader">First Name </td>
              <td>${FirstName} </td>
            </tr>
            <tr>
              <td class="CardHeader">Last Name</td>
              <td>${LastName} </td>
            </tr>
            <tr>
              <td class="CardHeader">Title
                </td>
              <td>${Title}
                </td>
            </tr>
            <tr>
              <td class="CardHeader">Country
                </td>
              <td>${Country}

```

```

        </td>
      </tr>
    </tbody>
  </table>
</td>
</tr>
</script>

<div id="container">
  <div id="Grid"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Function template

The function template approach is compatible with the strict guidelines of [Content-Security-Policy \(CSP\)](#). In the application, JavaScript functions with [string literals](#) can be used to add templates in the Syncfusion JavaScript (ES5) controls. This approach eliminates the need for the `unsafe-eval` keyword in the meta tag of project pages. It is essential to convert all inline string and script templates into function templates that comply with the [Content-Security-Policy \(CSP\)](#) guidelines in order to avoid potential security risks and enhance the overall safety of the application.

Lets discuss about converting the existing [string template](#) and [script template](#) into a function template approach.

Convert the existing string template into function template

To convert the existing [string template](#) into function template, create a function template and define the template using [template literals](#) enclosed in backticks. Add the following function template to the `index.html` file.

```
`js
```

```
var customTemplate = (data) => <div class="template">${data.ShipCountry}</div>
```

```
,
```

Here, the function template identifier (customTemplate) is assigned to the `template` property of the Grid control. Refer to the following code snippet.

INDEX.JS

```

//function template
var customTemplate = (data) => `<div
class="template">${data.ShipCountry}</div>`
var grid = new ej.grids.Grid({
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right', width:
125 },
    { field: 'CustomerName', headerText: 'Customer Name', width: 125 },
    { headerText: 'ShipCountry', template: customTemplate, width: 125 },

```

```

    ],
    dataSource: [
      { OrderID: 10248, ShipCountry: "France", CustomerName: "Paul Henriot" },
      { OrderID: 10249, ShipCountry: "Germany", CustomerName: "Karin Josephs" },
    ],
    { OrderID: 10250, ShipCountry: "Brazil", CustomerName: "Mario Pontes" },
    { OrderID: 10251, ShipCountry: "France", CustomerName: "Mary Saveley" }
  ],
  width: 'auto',
  height: 359
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container">
    <div id="Grid"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Convert the existing script template into function template

To convert the existing [script template](#) into function template, create a function template and define the template using [template literals](#) enclosed in backticks. Add the following function template to the `index.js` file.

```

`js
var customTemplate = (data) => `<tr>
<td class="photo">
${data.EmployeeID}

```

</td>

<td class="details">

First Name	\${data.FirstName}
Last Name	\${data.LastName}
Title	\${data.Title}
Country	\${data.Country}

</td>

</tr>

,

Here, the function template identifier (customTemplate) is assigned to the **template** property of the Grid control. Refer to the following code snippet.

INDEX.JS

```
var customTemplate = (data) => `<tr>
  <td class="photo">
    ${data.EmployeeID}
  </td>
  <td class="details">
    <table class="CardTable" cellpadding="3" cellspacing="2">
      <colgroup>
        <col width="50%">
        <col width="50%">
      </colgroup>
      <tbody>
        <tr>
          <td class="CardHeader">First Name </td>
          <td>${data.FirstName} </td>
        </tr>
        <tr>
          <td class="CardHeader">Last Name</td>
          <td>${data.LastName} </td>
        </tr>
        <tr>
          <td class="CardHeader">Title</td>
          <td>${data.Title}</td>
        </tr>
        <tr>
          <td class="CardHeader">Country</td>
          <td>${data.Country}</td>
        </tr>
      </tbody>
    </table>
  </td>
</tr>`;
var grid = new ej.grids.Grid({
  dataSource: employeeData,
```

```

    rowTemplate: customTemplate,
    columns: [
      { headerText: 'Employee ID', width: 150, textAlign: 'Center', field:
'OrderID' },
      { headerText: 'Employee Details', width: 300, field: 'EmployeeID' }
    ],
    height: 315
  });
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
  <script src="es5-datasource.js"></script>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container">
    <div id="Grid"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Getting started with Localization

Localization library allows you to localize the text content of the Syncfusion React UI Components. This is useful if you want to display the UI in a language other than English.

Loading translations

To load a translation object in your application, you can use the load function from the @syncfusion/ej2-base module. This function takes an object that contains the translations for various languages, with the keys being the language codes and the values being the translation objects.

INDEX.JS

```

ej.base.L10n.load({
  'de-DE': {

```

```

        'grid': {
            'EmptyRecord': 'Keine Aufzeichnungen angezeigt',
            'GroupDropArea': 'Ziehen Sie einen Spaltenkopf hier, um die
Gruppe ihre Spalte',
            'UnGroup': 'Klicken Sie hier, um die Gruppierung aufheben',
            'EmptyDataSourceError': 'DataSource darf bei der Erstaustlastung
nicht leer sein, da Spalten aus der dataSource im AutoGenerate
Spaltenraster',
            'Item': 'Artikel',
            'Items': 'Artikel'
        },
        'pager': {
            'currentPageInfo': '{0} von {1} Seiten',
            'totalItemsInfo': '({0} Beiträge)',
            'firstPageTooltip': 'Zur ersten Seite',
            'lastPageTooltip': 'Zur letzten Seite',
            'nextPageTooltip': 'Zur nächsten Seite',
            'previousPageTooltip': 'Zurück zur letzten Seit',
            'nextPagerTooltip': 'Zum nächsten Pager',
            'previousPagerTooltip': 'Zum vorherigen Pager'
        }
    });
ej.grids.Grid.Inject(ej.grids.Page, ej.grids.Group);
var grid = new ej.grids.Grid({
    dataSource: data,
    locale: 'de-DE',
    allowGrouping: true,
    allowPaging: true,
    pageSettings: { pageSize: 6 },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'ShipCity', headerText: 'Ship City', width: 150 },
        { field: 'ShipName', headerText: 'Ship Name', width: 150 }
    ],
    height: 220
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Changing current locale

The current locale can be changed for all the Syncfusion React UI Components in your application by invoking `setCulture` function with your desired culture name.

```

`ts
import {L10n, setCulture} from '@syncfusion/ej2-base';
L10n.load({
  'fr-BE': {
    'Button': {
      'id': 'Numéro de commande',
      'date': 'Date de commande'
    }
  }
})

```

```
});
setCulture('fr-BE');
`
```

Note: Before changing a culture globally, you need to ensure that locale text for the concerned culture is loaded through `L10n.load` function.

Internationalization

The `Internationalization` library provides support for formatting and parsing date and number objects using the official [Unicode CLDR](#) JSON data. The `en-US` locale is set as default *culture* and `USD` is set as default *currencyCode* for all Syncfusion JavaScript UI controls.

Loading culture data

It requires the following CLDR data to be load using `loadCldr` function for cultures other than `en-US`.

File Name	Path
ca-gregorian	cldr/main/en/ca-gregorian.json
timeZoneNames	cldr/main/en/timeZoneNames.json
numbers	cldr/main/en/numbers.json
numberingSystems	cldr/supplemental/numberingSystems.json
currencies	cldr/main/en/currencies.json

Note: For `en`, dependency files are already loaded in the library.

Installing CLDR data

CLDR data is available as npm package. So, we can install it through below command to our package.

```
`bash
npm install cldr-data
`
```

Binding to i18n library

```
`ts
import { loadCldr } from '@syncfusion/ej2-base';
loadCldr(enNumberData, entimeZoneData);
`
```

Changing global culture and currency code

To set the default culture and the currency code for all Syncfusion JavaScript UI controls, you can use the methods `setCulture` for setting the default locale and `setCurrencyCode` for setting the currency code.

```
`ts
import { setCulture, setCurrencyCode } from '@syncfusion/ej2-base';
setCulture('ar');
```

```
setCurrencyCode('QAR');
```

```
,
```

Note: If global culture is not set, then `en-US` is set as the default locale, and `USD` is set as the default currency code.

Manipulating numbers

```
<!-- markdownlint-disable MD024 -->
```

Supported format string

```
<!-- markdownlint-disable MD024 -->
```

Based on the [NumberFormatOptions](#) number formatting and parsing operations are processed. You need to specify some or all of the following properties mentioned below table.

No	Properties	Description
----	------------	-------------

---	---	---
-----	-----	-----

1	<code>format</code>	Denotes the format to be set .Possible values are
---	---------------------	---

2	<code>C</code>	denotes currency type.
---	----------------	------------------------

3	<code>P</code>	denotes percentage type.
---	----------------	--------------------------

E.g:	<code>formatNumber(1234344,{format:'N4'})</code>	
------	--	--

Note:	If no format is specified it takes numeric as default format type.
-------	--

2	<code>minimumFractionDigits</code>	Indicates the minimum number of fraction digits. Possible values are 0 to 20.
---	------------------------------------	---

3	<code>maximumFractionDigits</code>	Indicates the maximum number of fraction digits. Possible values are 0 to 20.
---	------------------------------------	---

4	<code>minimumSignificantDigits</code>	Indicates the minimum number of significant digits. Possible values are 1 to 21. Note: If <code>minimumSignificantDigits</code> is given it is mandatory to give <code>maximumSignificantDigits</code>
---	---------------------------------------	---

5	<code>maximumSignificantDigits</code>	Indicates the maximum number of significant digits. . Possible values are 1 to 21. Note: If <code>maximumSignificantDigits</code> is given it is mandatory to give <code>minimumSignificantDigits</code>
---	---------------------------------------	--

6	<code>useGrouping</code>	Indicates whether to enable the group separator or not . By default grouping value will be true.
---	--------------------------	--

7	<code>minimumIntegerDigits</code>	Indicates the minimum number of the integer digits to be placed in the value. Possible values are 1 to 21.
---	-----------------------------------	--

8	<code>currency</code>	Indicates the currency code which needs to considered for the currency formatting.
---	-----------------------	--

Note: The `minimumIntegerDigits`, `minimumFractionDigits` and `maximumFractionDigits` are categorized

as group one,

`minimumSignificantDigits` and `maximumSignificantDigits` are categorized as group two.

If group two properties are defined, then the group one properties will be ignored.

Custom number formatting and parsing

Custom number formatting and parsing can also be achieved by directly specifying the pattern in the **format** property of `NumberFormatOptions`. One or more of the custom format specifiers listed in the table below can be used to create custom number format.

Specifier	Description	Input	Format Output
0	Replaces the zero with the corresponding digit if one is present. Otherwise, zero appears in the result string.	<code>instance.formatNumber(123,{format: '0000' })</code>	<code>'0123'</code>
#	Replaces the "#" symbol with the corresponding digit if one is present; otherwise, no digit appears in the result string.	<code>instance.formatNumber(1234,{format: '####' })</code>	<code>'1234'</code>
.	Denotes the number of digits allowed after the decimal points if it's not specified then no need to specify decimal point values.	<code>instance.formatNumber(546321,{format: '###0.##0#' })</code>	<code>'546321.000'</code>
%	Percent specifier denotes the percentage type format.	<code>instance.formatNumber(1,{format: '0000 %' })</code>	<code>'0100 %'</code>
\$	Denotes the currency type format based on the global currency code specified.	<code>instance.formatNumber(13,{format: '\$ ###.00' })</code>	<code>'\$ 13.00'</code>
;	Denotes separate formats for positive, negative and zero values.	<code>instance.formatNumber(-120,{format: '###.##;(###.00);-0'})</code>	<code>'(120.00)'</code>
'String' (single Quotes)	Denotes the characters enclosed within single Quote(') to be replaced in the resultant string.	<code>instance.formatNumber(-123.44,{format: "####.## '@'"})</code>	<code>'123.44 @'</code>

Note: If a custom format pattern is specified, other [NumberFormatOptions](#) properties will not be considered.

Number Parsing``getNumberParser``

The [getNumberParser](#) method, which will return a function that parses a given string based on the [NumberFormatOptions](#) specified.

INDEX.JS

```
var intl = new ej.base.Internationalization();
var nParser = intl.getNumberParser({ format: 'P2' , useGrouping: false});
var val = nParser('123567.45%');
document.querySelector('.result').innerHTML = val + ' %';
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Form Validator</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
```

```

    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>FormattedValue:<span class="format text">123567.45%</span></div>
        <div>ParsedOutput:<span class="result text"> </span></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

``parseNumber``

The [parseNumber](#) method, which takes two arguments, the string value and [NumberFormatOptions](#) and returns the numeric value.

INDEX.JS

```

var intl = new ej.base.Internationalization();
var val = intl.parseNumber('$01,234,545.650', { format: 'C3', currency:
'USD', minimumIntegerDigits: 8 });
document.querySelector('.result').innerHTML = val + ' ';

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Form Validator</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">

```

```

        <div>FormattedValue:<span class="format
text">$01,234,545.650</span></div>
        <div>ParsedOutput:<span class="result text"> </span></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Number formatting

`getNumberFormat`

The [getNumberFormat](#) method will return a function that formats a given number based on the [NumberFormatOptions](#) specified.

INDEX.JS

```

var intl = new ej.base.Internationalization();
var nFormatter = intl.getNumberFormat({ skeleton: 'C3', currency:
'USD',minimumIntegerDigits:8});
var formattedValue = nFormatter(1234545.65)
document.querySelector('.result').innerHTML = formattedValue;

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Form Validator</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>Value:<span class="format text">1234545.65</span></div>
        <div>Formatted Value:<span class="result text"> </span></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>

```

```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

`formatNumber`

The [formatNumber](#) method, which takes two arguments, a numeric value and [NumberFormatOptions](#) and returns the formatted string.

INDEX.JS

```
var intl = new ej.base.Internationalization();
var formattedString = intl.formatNumber(12345.65, { format: 'C5' ,
useGrouping: false,
  minimumSignificantDigits:1, maximumSignificantDigits:3 });
document.querySelector('.result').innerHTML = formattedString;
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Form Validator</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div>Value:<span class="format text">1234545.65</span></div>
    <div>Formatted Value:<span class="result text"> </span></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Manipulating dateTime

Supported format string

Date formatting and parsing operations are performed based on the [DateFormatOptions](#). You need to specify some or all of the following properties mentioned in the table below.

| Options | Descriptions |

---	---	---
-----	-----	-----

| Type | It specifies the type of format to be used supported types .
1. **date**
 2. **dateTime**
 3. **time**
 Based on the type specified the supported skeletons are given below.
 1. short
 2. medium,
3. long
4. full
 E.g: formatDate(new Date(), {type: 'date', skeleton:medium})
 Note: If no type is specified then **date** type is set by default. |

skeleton	Specifies the format in which the dateTime format will process
----------	---

<!-- markdownlint-disable MD036 -->

Date type skeletons

skeleton	Option input	Format Output
----------	--------------	---------------

---	---	---
-----	-----	-----

short	{type: 'date', skeleton:'short'}	11/4/16
-------	----------------------------------	---------

medium	{type: 'date', skeleton:'medium'}	Nov 4, 2016
--------	-----------------------------------	-------------

long	{type: 'date', skeleton:'long'}	November 4, 2016
------	---------------------------------	------------------

full	{type: 'date', skeleton:full}	Friday, November 4, 2016
------	-------------------------------	--------------------------

Time type skeletons

skeleton	Option input	Format Output
----------	--------------	---------------

---	---	---
-----	-----	-----

short	{type: 'time', skeleton:'short'}	1:03 PM
-------	----------------------------------	---------

medium	{type: 'time', skeleton:'medium'}	1:03:04 PM
--------	-----------------------------------	------------

Long	{type: 'time', skeleton:'long'}	1:03:04 PM GMT+5
------	---------------------------------	------------------

full	{type: 'time', skeleton:'full'}	1:03:04 PM GMT+05:30
------	---------------------------------	----------------------

DateTime type skeletons

Skeleton	Option input	Format Output
----------	--------------	---------------

---	---	---
-----	-----	-----

short	{type: 'dateTime', skeleton:'short'}	11/4/16, 1:03 PM
-------	--------------------------------------	------------------

medium	{type: 'dateTime', skeleton:'medium'}	Nov 4, 2016, 1:03:04 PM
--------	---------------------------------------	-------------------------

Long	{type: 'dateTime', skeleton:'long'}	November 4, 2016 at 1:03:04 PM GMT+5
------	-------------------------------------	--------------------------------------

full	{type: 'dateTime', skeleton:'full'}	Friday, November 4, 2016 at 1:03:04 PM GMT+05:30
------	-------------------------------------	--

Additional skeletons

Apart from the standard date type formats additional format are supported by using the additional skeletons given in below table.

skeleton	Option input	Format Output
----------	--------------	---------------

---	---	---
-----	-----	-----

d	{skeleton:'d'}	7
---	----------------	---

E	{skeleton:'E'}	Mon
Ed	{skeleton:'Ed'}	7 Mon
Ehm	{skeleton:'Ehm'}	Mon 12:43 AM
EHm	{skeleton:'EHm;'}}	Mon 12:43
Ehms	{skeleton:'Ehms' }	Mon 2:45:23 PM
EHms	{skeleton:'EHms'}}	Mon 12:45:45
Gy	{skeleton:'Gy' }	2016 AD
GyMMM	{skeleton:'GyMMM'}	: Nov 2016 AD
GyMMMd	{skeleton:'GyMMMd'}	Nov 7, 2016 AD
GyMMMEd	{skeleton:'GyMMMEd'}	Mon, Nov 7, 2016 AD
h	{skeleton:'h'}	12 PM
H	{skeleton:'H'}	12
hm	{skeleton:'hm'}	12:59 PM
Hm	{skeleton:'Hm'}	12:59
hms	{skeleton:'hms'}	12:59:13 PM
Hms	{skeleton:'Hms'}	12:59:13
M	{skeleton:'M'}	11
Md	{skeleton:'Md'}	11/7
MEd	{skeleton:'hms'}	Mon, 11/7
MMM	{skeleton:'MMM'}	Nov
MMMEd	{skeleton:'MMMEd'}	Mon, Nov 7
MMMd	{skeleton:'MMMEd'}	Nov 7
ms	{skeleton:'ms'}	59:13
y	{skeleton:'y' }	2016
yM	{skeleton:'yM' }	11/2016
yMd	{skeleton:'yMd' }	11/7/2016
yMEd	{skeleton:'yMEd' }	Mon, 11/7/2016
yMMM	{skeleton:'yMMM' }	Nov 2016
yMMMd	{skeleton:'yMMMd' }	Nov 7, 2016
yMMMEd	{skeleton:'yMMMEd' }	Mon, Nov 7, 2016
yMMM	{skeleton:'yMMM' }	Nov 2016

Note: Culture specific format skeletons are also supported.

Custom formats

To use the custom date and time formats we need to specify the date/time pattern directly in the *format* property.

Custom format string must contain one or more of the following standard date/time symbols

Symbols	Description
-----	-----
G	Denotes the era in the date
y	Denotes the year.
M / L	Denotes month.
E / c	Denotes the day of week.
d	Denotes the day of month.
h / H	Denotes the hour. <i>h</i> for 12 hour and <i>H</i> for 24 hours format.
m	Denotes minutes.
s	Denotes seconds.
a	Denotes the am/pm designator it will only be displayed if hour is specified in the <i>h</i> format.
z	Denotes the time zone.
' (single quotes)	To display words in the formatted date you can specify the words with in the single quotes

Custom format example

```

`ts
import { Internationalization } from '@syncfusion/ej2-base';
let intl: Internationalization = new Internationalization();
let formattedString: string = intl.formatDate(new Date('1/12/2014 10:20:33'), { format: '\year:\y'
\month:\ MM' });
//Output: "year:2014 month:01"
`

```

Note: If the format property is given as an option, other properties are not considered.

<!-- markdownlint-enable MD036 -->

Date Parsing

`getDateParser`

The [getDateParser](#) method will return a function that parses a given string based on the [DateFormatOptions](#) specified.

INDEX.JS

```

var intl = new ej.base.Internationalization();
var dParser = intl.getDateParser({skeleton: 'full', type: 'dateTime'});
var val = dParser('Friday, November 4, 2016 at 1:03:04 PM GMT+05:30');

```

```
document.querySelector('.result').innerHTML = val.toString();
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Form Validator</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div>Formatted value:<span class="format text">Friday, November 4, 2016 at
1:03:04 PM GMT+05:30</span></div>
    <div>parsed Value:<span class="result text"> </span></div>

  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

`parseDate`

The date object is returned by the [parseDate](#) method, which takes two arguments, a string value and [DateFormatOptions](#).

INDEX.JS

```
var intl = new ej.base.Internationalization();
var val = intl.parseDate('11/2016',{skeleton: 'yM'});
document.querySelector('.result').innerHTML = val.toString();
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Form Validator</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
```

```

    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
    <div>Formatted value:<span class="format text">11/2016</span></div>
        <div>parsed Value:<span class="result text"> </span></div>

    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Date Formatting

`getDateFormat`

The [getDateFormat](#) method, which will return a function that formats a given date object based on the [DateFormatOptions](#) specified.

INDEX.JS

```

var intl = new ej.base.Internationalization();
var dFormatter = intl.getDateFormat({ skeleton: 'full', type: 'dateTime' });
var formattedString = dFormatter(new Date('1/12/2014 10:20:33'));
document.querySelector('.result').innerHTML = formattedString;

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Form Validator</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```

```

    <div id="container">
    <div>DateValue:<span class="format text">new Date('1/12/2014
10:20:33')</span></div>
    <div>Formatted Value:<span class="result text"> </span></div>

    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

`formatDate`

The [formatDate](#) method, which takes two arguments, the date object and [DateFormatOptions](#), returns the formatted string.

INDEX.JS

```

var intl = new ej.base.Internationalization();
var date = new Date();
var formattedString = intl.formatDate(new Date('1/12/2014 10:20:33'), {
    skeleton: 'GyMMM' });
document.querySelector('.result').innerHTML = formattedString;

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Form Validator</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
    <div>DateValue:<span class="format text">new Date('1/12/2014
10:20:33')</span></div>
    <div>Formatted Value:<span class="result text"> </span></div>

    </div>
</script>
var ele = document.getElementById('container');
if(ele) {

```

```

    ele.style.visibility = "visible";
  }
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

<!-- markdownlint-enable MD036 -->

Drag and drop in ##Platform_Name##

Drag and drop is a feature of a user interface that allows users to select an item or items and then move them to a different location or onto another interface element by "dragging" the selected item(s) with a pointing device (such as a mouse) and then "dropping" them at the desired location.

Syncfusion JavaScript controls support drag and drop feature through two libraries. These are [Draggable](#) and [Droppable](#).

Draggable

The Syncfusion's [Draggable](#) library allows users to make any DOM element draggable by passing it as a parameter to the [Draggable](#) constructor. This can be useful for creating interactive and user-friendly interfaces, such as allowing a user to reorder items in a list by dragging them. The below code snippet enables the draggable functionality for the specific DOM element passed to the [Draggable](#) constructor.

INDEX.JS

```

var dragElement = document.getElementById('element1');
var draggable = new ej.base.Draggable(dragElement, {clone: false});

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Draggable</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element1"><p>Draggable Element </p></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}

```

```

    </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Clone draggable element

Syncfusion provides the option to create a clone of a draggable element while the user is dragging it. It can be achieved by setting the [clone](#) property to `true`. Here's an example of how to create a clone of a draggable element.

INDEX.JS

```

var dragElement = document.getElementById('element1');
var draggable = new ej.base.Draggable(dragElement, {clone: false});

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Draggable</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element1"><p>Draggable Element </p></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Drag area

A drag area is a specific area within a user interface that is designated for drag and drop operations. When an object or element is dragged within a drag area, certain actions or events may be triggered. The user can specify the drag area by using the [dragArea](#) property. Refer to the below sample.

INDEX.JS

```

var draggable = new ej.base.Draggable(document.getElementById('element1'), {

```

```

        clone: false
    });
    var droppable = new ej.base.Droppable(document.getElementById('droppable'),
    {
        drop: (function (e) {
            e.droppedElement.querySelector('.drag-text').textContent =
            'Dropped';
        })
    });

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Draggable</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="droppable">
            <p class="drop"><span>Drop Target </span></p></div>
            <div id="element1"><p class="drag-text">Drag </p></div>
        </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Droppable

Droppable element refers to an area of a user interface that can receive a draggable element that is being moved by a user. Syncfusion's [Droppable](#) library converts any DOM element into a droppable zone, which accepts draggable elements.

When a draggable element is moved over a droppable element, the [drop](#) event can be triggered. The user can get details about the dropped element through the event argument. Based on the event argument, the user can append the dragged element to the target element.

Refer to the following code snippet to enable droppable zones.

INDEX.JS

```
var draggable = new ej.base.Draggable(document.getElementById('element1'), {
  clone: false
});
var droppable = new ej.base.Droppable(document.getElementById('droppable'), {
  drop: (function (e) {
    e.droppedElement.querySelector('.drag-text').textContent =
    'Dropped';
  })
});
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Draggable</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="droppable">
      <p class="drop"><span>Drop Target </span></p></div>
      <div id="element1"><p class="drag-text">Drag </p></div>
    </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

See also

- [Define handle element for draggable](#)
- [Restricting draggable within container](#)
- [Visual feedback of draggable element](#)
- [Accepting specific drag element in droppable](#)

Troubleshoot

Content Security Policy

Content Security Policy (CSP) is a security feature implemented by web browsers that helps to protect against attacks such as cross-site scripting (XSS) and data injection. It limits the sources from which content can be loaded on a web page.

To enable strict [Content Security Policy \(CSP\)](#), certain browser features are disabled by default. In order to use Syncfusion controls with strict CSP mode, it is essential to include following directives in the CSP meta tag.

- Syncfusion controls are rendered with calculated **inline styles** and **base64** font icons, which are blocked on a strict CSP-enabled site. To allow them, add the [style-src 'self' 'unsafe-inline'](#); and [font-src 'self' data:;](#) directives in the meta tag as follows.

HTML

```
<meta http-equiv="Content-Security-Policy" content="default-src 'self';  
style-src 'self' 'unsafe-inline';  
font-src 'self' data:;" />
```

- Syncfusion **material** and **tailwind** built-in themes contain a reference to the [Roboto's external font](#), which is also blocked. To allow them, add the [external font](#) reference to the [style-src](#) and [font-src](#) directives in the above meta tag.

The resultant meta tag is included within the `<head>` tag and resolves the CSP violation on the application's side when utilizing Syncfusion controls with material and tailwind themes.

HTML

```
<head>  
...  
<meta http-equiv="Content-Security-Policy" content="default-src 'self';  
style-src 'self' https://fonts.googleapis.com/ 'unsafe-inline';  
font-src 'self' https://fonts.googleapis.com/ https://fonts.gstatic.com/  
data:;" />  
</head>
```

Note: From the release 2023 Vol2 - 22.1 version, the Content Security Policy for Syncfusion controls has been enhanced by implementing a [function template](#) approach for template properties to eliminate the usage of the `unsafe-eval` directive in the CSP meta tag.

See also

- [How to resolve the Content Security Policy \(CSP\) errors](#)

How To

Updating Syncfusion npm packages

Keeping Syncfusion npm packages up to date is important to ensure that you have access to the latest features and bug fixes. The [npm-check-updates](#) package is a helpful tool that can be used to update your Syncfusion packages to their latest versions.

Updating All Syncfusion Packages

To update all Syncfusion packages, you can install the `npm-check-updates` package globally by running the following command,

```
`bash
npm install -g npm-check-updates
`
```

Next, use the `ncu` command to update the `package.json` file to the latest version for all `@syncfusion` packages,

```
`bash
ncu -u -f /^@syncfusion/
`
```

Finally, run the following commands to update the packages in `node_modules` and remove any duplicate packages that have been installed,

```
`bash
npm update
npm dedupe
`
```

Updating a specific npm package

You can also update a specific npm package by running the following commands from the command prompt in the root of your application,

```
`bash
npm update @syncfusion/ej2-grids
npm dedupe
`
```

This will update the specific package you have provided and run `npm dedupe` command which will remove any duplicate package.

How to load culture files in Essential JS 2 using Ajax

Ajax post can be used to load the `cldr` JSON files. To get started, install the `cldr-data` package using the following command.

```
`
npm install cldr-data
`
```

,

Loading culture using Ajax

To load locale files based on the selected culture, you can use the Ajax post method as shown in the following code snippet. The cldr files will be loaded based on the selected culture using the following function.

,

```
//Function for loading locale files based on culture name
function loadCultureFiles(name:any) {
  let files: string[] = ['ca-gregorian.json', 'numbers.json', 'timeZoneNames.json'];
  let loadCulture = function (prop:any) {
    let val:string, ajax: Ajax;
    if (name === 'ar' && prop === files.length - 1) {
      ajax = new Ajax( './node_modules/cldr-data/supplemental/' + files[prop], 'GET', false);
    } else {
      ajax = new Ajax( './node_modules/cldr-data/main/' + name + '/' + files[prop], 'GET', false);
    }
    ajax.onSuccess = function (value:any) {
      val = value;
      loadCldr(JSON.parse(val));
    };
    ajax.send();
  };
  for (let prop = 0; prop < files.length; prop++) {
    loadCulture(prop);
  }
}
loadCultureFiles('de');
```

,

Loading translations

To load the translation object in your application, use the `load` function of the `L10n` class.

,

```
//loading locale files
L10n.load({
  'de': {
```

```
'calendar': { today: 'heute' },
},
});
`
```

Loading CLDR files using loadCldr function

Pass the loaded locale files into the `loadCldr` function, as shown in the following example. The `setCulture` method allows you to set the required culture.

```
`
import { Calendar } from '@syncfusion/ej2-calendars';
//import the loadCldr from ej2-base
import { loadCldr, L10n, Ajax, setCulture } from '@syncfusion/ej2-base';
declare var require: any;
//loading locale files
L10n.load({
  'de': {
    'calendar': { today: 'heute' },
  },
});
//Function for loading locale files based on culture name
function loadCultureFiles(name:any) {
  let files: string[] = ['ca-gregorian.json', 'numbers.json', 'timeZoneNames.json'];
  let loadCulture = function (prop:any) {
    let val:string, ajax: Ajax;
    if (name === 'ar' && prop === files.length - 1) {
      ajax = new Ajax( './node_modules/cldr-data/supplemental/' + files[prop], 'GET', false);
    } else {
      ajax = new Ajax( './node_modules/cldr-data/main/' + name + '/' + files[prop], 'GET', false);
    }
    ajax.onSuccess = function (value:any) {
      val = value;
      loadCldr(JSON.parse(val));
    };
    ajax.send();
  };
}
```

```

for (let prop = 0; prop < files.length; prop++) {
  loadCulture(prop);
}
}

loadCultureFiles('de');

let calendarObject: Calendar = new Calendar({
  //sets the locale.
  locale: 'de'
});

calendarObject.appendTo('#element');
setCulture('de');calendarObject.appendTo('#element');
setCulture('de');
`

```

All components' localized texts are provided in the ej2-locale [GitHub](#) repository. You can also find examples in this [link](#).

How to load the culture file in Essential JS 2

In Essential JS 2, the culture file can be loaded by using the loadCldr function. The culture file contains information about the specific culture, such as date and time formats, currency symbols, and translations for UI elements.

Installing the CLDR JSON Files

To use the loadCldr function, you first need to install the CLDR JSON files from the npm package. You can do this by running the following command.

```

npm install cldr-data
`

```

Loading the culture

To load the culture file in your application, you need to import the required locale and supplemental files from the `cldr` package. For example, to load the German culture, you would import the following files.

```

import * as n1 from '../node_modules/cldr-data/main/de/currencies.json'
import * as n2 from '../node_modules/cldr-data/main/de/timeZoneNames.json';
import * as n3 from '../node_modules/cldr-data/main/de/numbers.json';
import * as n4 from '../node_modules/cldr-data/main/de/ca-gregorian.json';
import * as s from '../node_modules/cldr-data/supplemental/currencyData.json';
import * as s2 from '../node_modules/cldr-data/supplemental/numberingSystems.json';
`

```

Loading translations

To load the translation object in the application, use the `load` function of the `L10n` class. For example, the following code loads a German translation for the word "today" in the calendar component.

```
L10n.load({
  'de': {
    'calendar': { today: 'heute' }
  }
});
```

Using the loadCldr function

Once you have imported the necessary culture files, you can use the `loadCldr` function to load them in your application. You also need to pass the culture code in the `setCulture` function.

```
import { Calendar } from '@syncfusion/ej2-calendars';
import { Tab } from '@syncfusion/ej2-navigations';
import { L10n, setCulture, loadCldr } from '@syncfusion/ej2-base';
import * as n1 from '../node_modules/cldr-data/main/de/currencies.json';
import * as n2 from '../node_modules/cldr-data/main/de/timeZoneNames.json';
import * as n3 from '../node_modules/cldr-data/main/de/numbers.json';
import * as n4 from '../node_modules/cldr-data/main/de/ca-gregorian.json';
import * as s from '../node_modules/cldr-data/supplemental/currencyData.json';
import * as s2 from '../node_modules/cldr-data/supplemental/numberingSystems.json';

L10n.load({
  'de': {
    'calendar': { today: 'heute' }
  }
});

//Initialize calendar component.
let calendarObject: Calendar = new Calendar();

//Render initialized calendar.
calendarObject.appendTo('#element');

loadCldr(n1, n2, n3, n4, s, s2);
```

```
setCulture('de');
```

```
,
```

All components' localized texts are provided in the ej2-locale [GitHub](#) repository. Also find the example sample from this [link](#).

How to resolve Content Security Policy (CSP) errors

Enabling the strict Content Security Policy (CSP) may cause the following issues with the Essential JS 2 controls in your application.

Template rendering

From the 2023 Vol2 - 22.1 release onwards, the Content Security Policy for Syncfusion controls has been enhanced. The usage of the `unsafe-eval` directive has been eliminated from the CSP meta tag.

Note: If users prefer to continue using inline string and external templates, it is necessary to include the `unsafe-eval` directive in the CSP meta tag in order to bypass the CSP violation.

Image loading

3D Chart

Es5 getting started in EJ2 JavaScript 3D Chart control

This section explains you the steps required to create a simple 3D Chart and demonstrate the basic usage of the 3D Chart control.

Dependencies

Below is the list of minimum dependencies required to use the 3D Chart.

```
|-- @syncfusion/ej2-charts
|-- @syncfusion/ej2-base
|-- @syncfusion/ej2-data
|-- @syncfusion/ej2-pdf-export
|-- @syncfusion/ej2-file-utils
|-- @syncfusion/ej2-compression
|-- @syncfusion/ej2-svg-base
```

```
,
```

Setup for local environment

Refer the following steps for setup your local environment.

Step 1: Create a root folder **myapp** for your application.

Step 2: Create **myapp/resources** folder to store local scripts and styles files.

Step 3: Create **myapp/index.js** and **myapp/index.html** files for initializing Essential JS 2 3D Chart control.

Adding Syncfusion resources

The Essential JS 2 3D Chart control can be initialized by using either of the following ways.

- Using local script.
- Using CDN link for script.

Using local script

You can get the global scripts and styles from the [Essential Studio JavaScript \(Essential JS 2\)](#) build installed location.

After installing the Essential JS 2 product build, you can copy the chart and its dependencies scripts and style file into the **resources/scripts** and **resources/styles** folder.

Refer the below code to find location chart's script and style file.

Syntax:

Dependency script: **(installed location)/Syncfusion/Essential Studio/{RELEASEVERSION}/Essential JS 2/{DEPENDENCYPACKAGENAME}/dist/global/{DEPENDENCYPACKAGE_NAME}.min.js**

Script: **(installed location)/Syncfusion/Essential Studio/{RELEASEVERSION}/Essential JS 2/{PACKAGENAME}/dist/global/{PACKAGE_NAME}.min.js**

Example:

Dependency script: **C:/Program Files (x86)/Syncfusion/Essential Studio/15.4.30/Essential JS 2/ej2-base/dist/global/ej2-base.min.js**

Script: **C:/Program Files (x86)/Syncfusion/Essential Studio/15.4.30/Essential JS 2/ej2-charts/dist/global/ej2-charts.min.js**

After copying the files, then you can refer the chart's scripts into the **index.html** file. The below html code example shows the minimal dependency of chart.

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Essential JS 2 3D Chart</title>
<!-- Essential JS 2 Chart's dependent scripts -->
<!-- Essential JS 2 Chart's global script -->
</head>
<body>
</body>
</html>
```

,

Using CDN link for script

Using CDN link, you can directly refer the chart control's script into the `index.html`.

Refer the chart's CDN links as below

Syntax:

Script: `http://cdn.syncfusion.com/ej2/{PACKAGENAME}/dist/global/{PACKAGENAME}.min.js`

Example:

Script: <http://cdn.syncfusion.com/ej2/ej2-charts/dist/global/ej2-charts.min.js>

The below html code example shows the minimal dependency of chart.

```
`html
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Essential JS 2 3D Chart</title>
<!-- Essential JS 2 Chart's global script -->
</head>
<body>
</body>
</html>
```

,

Adding 3D Chart control

Now, you can start adding 3D Chart control in the application. For getting started, add a **div** element for 3D Chart control in `index.html`. Then refer the `index.js` file into the `index.html` file.

In this document context we are going to use `ej2.min.js` which includes all the Essential JS 2 components and its dependent scripts.

```
`html
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Essential JS 2 3D Chart</title>
<!-- Essential JS 2 all script -->
</head>
<body>
<!-- Add the HTML <div> element for 3D Chart -->
```

```
<div id="element"></div>
</body>
</html>
`
```

Place the following 3D Chart code in the **index.js**.

```
var chart3D = new ej.charts.Chart3D();
chart3D.appendTo('#element');
`
```

The below example shows a basic 3D Chart.

INDEX.JS

```
var chart3D = new ej.charts.Chart3D();
chart3D.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Populate 3D Chart with data

This section explains how to plot below JSON data to the 3D Chart.

```
var chartData = [
```

```
{ month: 'Jan', sales: 35 }, { month: 'Feb', sales: 28 },
{ month: 'Mar', sales: 34 }, { month: 'Apr', sales: 32 },
{ month: 'May', sales: 40 }, { month: 'Jun', sales: 32 },
{ month: 'Jul', sales: 35 }, { month: 'Aug', sales: 55 },
{ month: 'Sep', sales: 38 }, { month: 'Oct', sales: 30 },
{ month: 'Nov', sales: 25 }, { month: 'Dec', sales: 32 }
];
`
```

Add a series object to the 3D Chart by using [series](#) property. Now map the field names `month` and `sales` in the JSON data to the [xName](#) and [yName](#) properties of the series, then set the JSON data to [dataSource](#) property.

Since the JSON contains category data, set the [valueType](#) for horizontal axis to `Category`. By default, the axis `valueType` is `Numeric`.

INDEX.JS

```
var chartData = [
    { month: 'Jan', sales: 35 }, { month: 'Feb', sales: 28 },
    { month: 'Mar', sales: 34 }, { month: 'Apr', sales: 32 },
    { month: 'May', sales: 40 }, { month: 'Jun', sales: 32 },
    { month: 'Jul', sales: 35 }, { month: 'Aug', sales: 55 },
    { month: 'Sep', sales: 38 }, { month: 'Oct', sales: 30 },
    { month: 'Nov', sales: 25 }, { month: 'Dec', sales: 32 }
];
var chart3D = new ej.charts.Chart3D({
    primaryXAxis: {
        valueType: 'Category'
    },
    series:[{
        // dataSource for 3d chart series
        dataSource: chartData,
        xName: 'month',
        yName: 'sales',
        type: 'Column'
    }]
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The sales data are in thousands, so format the vertical axis label by adding \$ as a prefix and K as a suffix to each label. This can be achieved by setting the ``${value}K`` to the [labelFormat](#) property of axis. Here, `{value}` act as a placeholder for each axis label.

INDEX.JS

```

var chartData = [
    { month: 'Jan', sales: 35 }, { month: 'Feb', sales: 28 },
    { month: 'Mar', sales: 34 }, { month: 'Apr', sales: 32 },
    { month: 'May', sales: 40 }, { month: 'Jun', sales: 32 },
    { month: 'Jul', sales: 35 }, { month: 'Aug', sales: 55 },
    { month: 'Sep', sales: 38 }, { month: 'Oct', sales: 30 },
    { month: 'Nov', sales: 25 }, { month: 'Dec', sales: 32 }
];
var chart3D = new ej.charts.Chart3D({
    primaryXAxis: {
        valueType: 'Category'
    },
    primaryYAxis: {
        // label format for axis
        labelFormat: `${value}K`
    },
    series:[{
        dataSource: chartData,
        xName: 'month',
        yName: 'sales',
        type: 'Column'
    }]
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">

```

```

<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Add 3D Chart title

You can add a title using [title](#) property to the 3D Chart to provide quick information to the user about the data plotted in the 3D Chart.

INDEX.JS

```

var chartData = [
    { month: 'Jan', sales: 35 }, { month: 'Feb', sales: 28 },
    { month: 'Mar', sales: 34 }, { month: 'Apr', sales: 32 },
    { month: 'May', sales: 40 }, { month: 'Jun', sales: 32 },
    { month: 'Jul', sales: 35 }, { month: 'Aug', sales: 55 },
    { month: 'Sep', sales: 38 }, { month: 'Oct', sales: 30 },
    { month: 'Nov', sales: 25 }, { month: 'Dec', sales: 32 }
];
var chart3D = new ej.charts.Chart3D({
    // Title for 3d chart
    title: 'Sales Analysis',
    primaryXAxis: {
        valueType: 'Category',
        title: 'Month'
    },
    primaryYAxis: {
        labelFormat: '${value}K'
    },
    series:[{
        dataSource: chartData,
        xName: 'month',
        yName: 'sales',
        type: 'Column'
    }],
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Enable legend

You can use legend for the 3D Chart by setting the [visible](#) property to true in [legendSettings](#) object.

INDEX.JS

```

var chartData = [
  { month: 'Jan', sales: 35 }, { month: 'Feb', sales: 28 },
  { month: 'Mar', sales: 34 }, { month: 'Apr', sales: 32 },
  { month: 'May', sales: 40 }, { month: 'Jun', sales: 32 },
  { month: 'Jul', sales: 35 }, { month: 'Aug', sales: 55 },
  { month: 'Sep', sales: 38 }, { month: 'Oct', sales: 30 },
  { month: 'Nov', sales: 25 }, { month: 'Dec', sales: 32 }
];
var chart3D = new ej.charts.Chart3D({
  // Legend for 3d chart
  legendSettings: {
    visible: true
  },
  primaryXAxis: {
    valueType: 'Category'
  },
  primaryYAxis: {
    labelFormat: '${value}K'
  },
  series:[{
    dataSource: chartData,
    name: 'Sales',
    xName: 'month',

```

```

        yName: 'sales',
        type: 'Column'
    }],
    title: 'Sales Analysis'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Add data label

You can add data labels to improve the readability of the 3D Chart. This can be achieved by setting the [visible](#) property to true in the [dataLabel](#) object. Now, the data labels are arranged smartly based on series.

INDEX.JS

```

var chartData = [
  { month: 'Jan', sales: 35 }, { month: 'Feb', sales: 28 },
  { month: 'Mar', sales: 34 }, { month: 'Apr', sales: 32 },
  { month: 'May', sales: 40 }, { month: 'Jun', sales: 32 },
  { month: 'Jul', sales: 35 }, { month: 'Aug', sales: 55 },
  { month: 'Sep', sales: 38 }, { month: 'Oct', sales: 30 },
  { month: 'Nov', sales: 25 }, { month: 'Dec', sales: 32 }
];
var chart3D = new ej.charts.Chart3D({
  primaryXAxis: {
    valueType: 'Category'
  },

```



```

primaryYAxis: {
    labelFormat: '${value}K'
},
series:[{
    dataSource: chartData,
    xName: 'month',
    name: 'Sales',
    yName: 'sales',
    type: 'Column',
    // Data label for 3d chart series
    dataLabel: {
        visible: true
    }
}],
legendSettings: { visible: true },
title: 'Sales Analysis'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Enable tooltip

The tooltip is useful when you cannot display information by using the data labels due to space constraints. You can enable tooltip by setting the [enable](#) property as true in [tooltip](#) object.

INDEX.JS

```

var chartData = [

```

```

        { month: 'Jan', sales: 35 }, { month: 'Feb', sales: 28 },
        { month: 'Mar', sales: 34 }, { month: 'Apr', sales: 32 },
        { month: 'May', sales: 40 }, { month: 'Jun', sales: 32 },
        { month: 'Jul', sales: 35 }, { month: 'Aug', sales: 55 },
        { month: 'Sep', sales: 38 }, { month: 'Oct', sales: 30 },
        { month: 'Nov', sales: 25 }, { month: 'Dec', sales: 32 }
    ];
    var chart3D = new ej.charts.Chart3D({
        // Tooltip for 3d chart
        tooltip: {
            enable: true
        },
        primaryXAxis: {
            valueType: 'Category'
        },
        primaryYAxis: {
            labelFormat: '${value}K'
        },
        series:[{
            dataSource: chartData,
            name:'Sales',
            xName: 'month',
            yName: 'sales',
            type: 'Column',
            dataLabel: {
                visible: true
            }
        }
    ]],
    legendSettings: { visible: true },
    title: 'Sales Analysis'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {

```

```

    ele.style.visibility = "visible";
  }
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

<!-- markdownlint-disable MD036 -->

Working with data in EJ2 JavaScript 3D Chart control

Local data

A simple JSON data can be bound to the 3D chart using [dataSource](#) property in series. Now map the fields in JSON to [xName](#) and [yName](#) properties.

INDEX.JS

```

var chartData = [
  { month: 'Jan', sales: 35 }, { month: 'Feb', sales: 28 },
  { month: 'Mar', sales: 34 }, { month: 'Apr', sales: 32 },
  { month: 'May', sales: 40 }, { month: 'Jun', sales: 32 },
  { month: 'Jul', sales: 35 }, { month: 'Aug', sales: 55 },
  { month: 'Sep', sales: 38 }, { month: 'Oct', sales: 30 },
  { month: 'Nov', sales: 25 }, { month: 'Dec', sales: 32 }
];
var chart3D = new ej.charts.Chart3D({
  primaryXAxis: {
    valueType: 'Category',
    labelRotation: -45,
    labelPlacement: 'BetweenTicks'
  },
  series:[{
    dataSource: chartData,
    type: 'Column',
    xName: 'month',
    yName: 'sales'
  }],
  wallColor: 'transparent',
  enableRotation: true,
  rotation: 7,
  tilt: 10,
  depth: 100,
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>

```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Remote data

The remote data can be bound to the 3D chart using the [DataManager](#). The [DataManager](#) requires minimal information like web service URL, adaptor and cross domain to interact with service endpoint properly. Assign the instance of the [DataManager](#) to the [dataSource](#) property in series and map the fields of data to [xName](#) and [yName](#) properties. You can also use the [query](#) property of the series to filter the data.

INDEX.JS

```

var dataManager = new ej.data.DataManager({
    url: 'https://services.syncfusion.com/js/production/api/orders'
});
var query = new ej.data.Query().take(5).where('Estimate', 'lessThan', 3,
false);
var chart3D = new ej.charts.Chart3D({
    primaryXAxis: {
        valueType: 'Category',
        labelRotation: -45,
        labelPlacement: 'BetweenTicks'
    },
    series:[{
        dataSource: dataManager,
        type: 'Column',
        xName: 'CustomerID', yName: 'Freight', query: query
    }],
    wallColor: 'transparent',
    enableRotation: true,
    rotation: 7,
    tilt: 10,
    depth: 100,
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

```

```

<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Binding data using ODataAdaptor

OData is a standardized protocol for creating and consuming data. You can retrieve data from OData service using the **DataManager**. Refer to the following code example for remote data binding using OData service.

INDEX.JS

```

var query = new ej.data.Query();
var data = new ej.data.DataManager({
    url: 'https://services.syncfusion.com/js/production/api/orders',
    adaptor: new ej.data.ODataAdaptor()
});
var chart3D = new ej.charts.Chart3D({
    primaryXAxis: {
        valueType: 'Category',
        labelRotation: -45,
        labelPlacement: 'BetweenTicks'
    },
    series:[{
        dataSource: dataManager,
        type: 'Column',
        xName: 'CustomerID', yName: 'Freight', query: query,
    }],
    wallColor: 'transparent',
    enableRotation: true,
    rotation: 7,
    tilt: 10,
    depth: 100,
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Empty points

The data points that uses the **null** or **undefined** as value are considered as empty points. The empty data points are ignored and is not plotted in the chart. When the data is provided by using the points property, by using [emptyPointSettings](#) property in series, the empty can be customized. The default [mode](#) of the empty point is **Gap**.

INDEX.JS

```

var chartData = [
  { month: 'Jan', sales: 35 }, { month: 'Feb', sales: 28 },
  { month: 'Mar', sales: null }, { month: 'Apr', sales: 32 },
  { month: 'May', sales: 40 }, { month: 'Jun', sales: 32 },
  { month: 'Jul', sales: 35 }, { month: 'Aug', sales: undefined },
  { month: 'Sep', sales: 38 }, { month: 'Oct', sales: 30 },
  { month: 'Nov', sales: 25 }, { month: 'Dec', sales: 32 }
];
var chart3D = new ej.charts.Chart3D({
  primaryXAxis: {
    valueType: 'Category',
    labelRotation: -45,
    labelPlacement: 'BetweenTicks'
  },
  series:[{
    dataSource: chartData,
    xName: 'month',
    yName: 'sales',
    type: 'Column',

```

```

        emptyPointSettings: {
            mode: 'Gap'
        }
    }],
    wallColor: 'transparent',
    enableRotation: true,
    rotation: 7,
    tilt: 10,
    depth: 100,
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Customizing empty point

The specific color for empty point can be set by the [fill](#) property in [emptyPointSettings](#).

INDEX.JS

```

var chartData = [
  { month: 'Jan', sales: 35 }, { month: 'Feb', sales: 28 },
  { month: 'Mar', sales: null }, { month: 'Apr', sales: 32 },
  { month: 'May', sales: 40 }, { month: 'Jun', sales: 32 },
  { month: 'Jul', sales: 35 }, { month: 'Aug', sales: undefined },
  { month: 'Sep', sales: 38 }, { month: 'Oct', sales: 30 },
  { month: 'Nov', sales: 25 }, { month: 'Dec', sales: 32 }
];
var chart3D = new ej.charts.Chart3D({

```

```

    primaryXAxis: {
      valueType: 'Category',
      labelRotation: -45,
      labelPlacement: 'BetweenTicks'
    },
    series:[{
      dataSource: chartData,
      xName: 'month',
      yName: 'sales',
      type: 'Column',
      emptyPointSettings: {
        mode: 'Average',
        fill: 'green'
      }
    }],
    wallColor: 'transparent',
    enableRotation: true,
    rotation: 7,
    tilt: 10,
    depth: 100,
  }, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```


Dimensions in EJ2 JavaScript 3D Chart control

Size for container

The 3D chart can be rendered to its container size and it can be set via inline or CSS as demonstrated below.

```
<div id='container'>

<div id='element' style="width:650px; height:350px;"></div>

</div>
`
```

INDEX.JS

```
var chartData = [
  { month: 'Jan', sales: 35 }, { month: 'Feb', sales: 28 },
  { month: 'Mar', sales: 34 }, { month: 'Apr', sales: 32 },
  { month: 'May', sales: 40 }, { month: 'Jun', sales: 32 },
  { month: 'Jul', sales: 35 }, { month: 'Aug', sales: 55 },
  { month: 'Sep', sales: 38 }, { month: 'Oct', sales: 30 },
  { month: 'Nov', sales: 25 }, { month: 'Dec', sales: 32 }
];
var chart3D = new ej.charts.Chart3D({
  primaryXAxis: {
    valueType: 'Category',
  },
  series: [
    {
      dataSource: chartData, xName: 'month', yName: 'sales',
      type: 'Column'
    }
  ],
  enableRotation: true,
  rotation: 22,
  depth: 100,
  width: '650px',
  height: '350px'
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>
```

```

    <div id="container">
      <div id="element"></div>
    </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

<!-- markdownlint-disable MD036 -->

Size for chart

<!-- markdownlint-disable MD036 -->

The size of the 3D chart can be set directly through [width](#) and [height](#) properties.

In Pixel

The size of the 3D chart can be set in pixel as demonstrated below.

INDEX.JS

```

var chartData = [
  { month: 'Jan', sales: 35 }, { month: 'Feb', sales: 28 },
  { month: 'Mar', sales: 34 }, { month: 'Apr', sales: 32 },
  { month: 'May', sales: 40 }, { month: 'Jun', sales: 32 },
  { month: 'Jul', sales: 35 }, { month: 'Aug', sales: 55 },
  { month: 'Sep', sales: 38 }, { month: 'Oct', sales: 30 },
  { month: 'Nov', sales: 25 }, { month: 'Dec', sales: 32 }
];
var chart3D = new ej.charts.Chart3D({
  primaryXAxis: {
    valueType: 'Category',
  },
  series: [
    {
      dataSource: chartData, xName: 'month', yName: 'sales',
      type: 'Column'
    }
  ],
  enableRotation: true,
  rotation: 22,
  depth: 100,
  width: '650', height: '350'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">

```

```

<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

In Percentage

By setting the value in percentage, 3D chart gets its dimension with respect to its container. For example, when the height is **50%**, chart renders to half of the container height.

INDEX.JS

```

var chartData = [
    { month: 'Jan', sales: 35 }, { month: 'Feb', sales: 28 },
    { month: 'Mar', sales: 34 }, { month: 'Apr', sales: 32 },
    { month: 'May', sales: 40 }, { month: 'Jun', sales: 32 },
    { month: 'Jul', sales: 35 }, { month: 'Aug', sales: 55 },
    { month: 'Sep', sales: 38 }, { month: 'Oct', sales: 30 },
    { month: 'Nov', sales: 25 }, { month: 'Dec', sales: 32 }
];
var chart3D = new ej.charts.Chart3D({
    primaryXAxis: {
        valueType: 'Category',
    },
    series: [
        {
            dataSource: chartData, xName: 'month', yName: 'sales',
            type: 'Column'
        }
    ],
    enableRotation: true,
    rotation: 22,
    depth: 100,
    width: '80%', height: '90%'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Note: When you do not specify the size, it takes 450px as the height and window size as its width.

Category axis in EJ2 JavaScript 3D Chart control

The category axis is the horizontal axis of a 3D chart that shows text values rather than numerical values. Compared to the vertical axis, this axis has fewer labels. The following sample shows to render the 3D chart using category axis.

INDEX.JS

```

var chartData = [
  { country: "USA", gold: 50, silver: 70, bronze: 45 },
  { country: "China", gold: 40, silver: 60, bronze: 55 },
  { country: "Japan", gold: 70, silver: 60, bronze: 50 },
  { country: "Australia", gold: 60, silver: 56, bronze: 40 },
  { country: "France", gold: 50, silver: 45, bronze: 35 },
  { country: "Germany", gold: 40, silver: 30, bronze: 22 },
  { country: "Italy", gold: 40, silver: 35, bronze: 37 },
  { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart3D = new ej.charts.Chart3D({
  primaryXAxis: {
    valueType: 'Category',
  },
  series:[{
    dataSource: chartData,
    xName: 'country', yName: 'gold',
    // Series type as column series
    type: 'Column'
  }
]);

```

```

    }],
    title: 'Olympic Medals',
    wallColor: 'transparent',
    enableRotation: true,
    rotation: 7,
    tilt: 10,
    depth: 100
  }, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Note: To use category axis, we need to inject **Category3D** module using **Chart3D.Inject(Category3D)** method and set the [valueType](#) of axis to **Category**.

Labels placement

By default, category axis labels are placed between ticks in an axis. It can also be placed on ticks using the [labelPlacement](#) property.

INDEX.JS

```

var chartData = [
  { country: "USA", gold: 50, silver: 70, bronze: 45 },
  { country: "China", gold: 40, silver: 60, bronze: 55 },
  { country: "Japan", gold: 70, silver: 60, bronze: 50 },
  { country: "Australia", gold: 60, silver: 56, bronze: 40 },
  { country: "France", gold: 50, silver: 45, bronze: 35 },
  { country: "Germany", gold: 40, silver: 30, bronze: 22 },

```

```

    { country: "Italy", gold: 40, silver: 35, bronze: 37 },
    { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
  ];
  var chart3D = new ej.charts.Chart3D({
    primaryXAxis: {
      valueType: 'Category',
      labelPlacement: 'OnTicks',
    },
    series:[{
      dataSource: chartData,
      xName: 'country', yName: 'gold', type: 'Column'
    }],
    title: 'Olympic Medals',
    wallColor: 'transparent',
    enableRotation: true,
    rotation: 7,
    tilt: 10,
    depth: 100,
  }, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
  var ele = document.getElementById('container');
  if(ele) {
    ele.style.visibility = "visible";
  }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Range

The range of the category axis can be customized using [minimum](#), [maximum](#) and [interval](#) properties of the axis.

INDEX.JS

```

var chartData = [
  { country: "USA", gold: 50, silver: 70, bronze: 45 },
  { country: "China", gold: 40, silver: 60, bronze: 55 },
  { country: "Japan", gold: 70, silver: 60, bronze: 50 },
  { country: "Australia", gold: 60, silver: 56, bronze: 40 },
  { country: "France", gold: 50, silver: 45, bronze: 35 },
  { country: "Germany", gold: 40, silver: 30, bronze: 22 },
  { country: "Italy", gold: 40, silver: 35, bronze: 37 },
  { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart3D = new ej.charts.Chart3D({
  primaryXAxis: {
    valueType: 'Category',
    interval: 2, minimum: 1, maximum: 5
  },
  series:[{
    dataSource: chartData,
    xName: 'country', yName: 'gold', type: 'Column'
  }],
  title: 'Olympic Medals',
  wallColor: 'transparent',
  enableRotation: true,
  rotation: 7,
  tilt: 10,
  depth: 100,
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Indexed category axis

The category axis can also be rendered based on the index values of the data source. This can be achieved by defining the [isIndexed](#) property to **true** in the axis.

INDEX.JS

```
var chartData = [
  { country: "USA", gold: 50, silver: 70, bronze: 45 },
  { country: "China", gold: 40, silver: 60, bronze: 55 },
  { country: "Japan", gold: 70, silver: 60, bronze: 50 },
  { country: "Australia", gold: 60, silver: 56, bronze: 40 },
  { country: "France", gold: 50, silver: 45, bronze: 35 },
  { country: "Germany", gold: 40, silver: 30, bronze: 22 },
  { country: "Italy", gold: 40, silver: 35, bronze: 37 },
  { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart3D = new ej.charts.Chart3D({
  primaryXAxis: {
    valueType: 'Category',
    isIndexed: true
  },
  series:[{
    dataSource: chartData,
    xName: 'country', yName: 'gold', type: 'Column'
  },
  {
    dataSource: chartData,
    xName: 'country', yName: 'silver', type: 'Column'
  }],
  title: 'Olympic Medals',
  wallColor: 'transparent',
  enableRotation: true,
  rotation: 7,
  tilt: 10,
  depth: 100,
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>
```



```

    <div id="container">
      <div id="element"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

<!-- markdownlint-disable MD036 -->

Numeric axis in EJ2 JavaScript 3D Chart control

INDEX.JS

```

var chartData = [{ x: 1, y: 7 }, { x: 2, y: 1 }, { x: 3, y: 1 }, { x: 4, y: 14 }, { x: 5, y: 1 }, { x: 6, y: 10 }, { x: 7, y: 8 }, { x: 8, y: 6 }, { x: 9, y: 10 }, { x: 10, y: 10 }, { x: 11, y: 16 }, { x: 12, y: 6 }, { x: 13, y: 14 }, { x: 14, y: 7 }, { x: 15, y: 5 }, { x: 16, y: 2 }, { x: 17, y: 14 }, { x: 18, y: 7 }, { x: 19, y: 7 }, { x: 20, y: 10 }];
var chart3D = new ej.charts.Chart3D({
  primaryXAxis: {
    valueType: 'Double',
  },
  series: [
    {
      type: 'Column', xName: 'x', yName: 'y', columnSpacing: 0.1,
      dataSource: chartData
    }
  ],
  wallColor: 'transparent',
  enableRotation: true,
  rotation: 7,
  tilt: 10,
  depth: 100
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>

```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Range

The range of an axis will be calculated automatically based on the provided data, and it can also be customized by using the [minimum](#), [maximum](#) and [interval](#) properties of the axis.

INDEX.JS

```

var chartData = [{ x: 1, y: 7 }, { x: 2, y: 1 }, { x: 3, y: 1 }, { x: 4, y:
14 }, { x: 5, y: 1 }, { x: 6, y: 10 },
    { x: 7, y: 8 }, { x: 8, y: 6 }, { x: 9, y: 10 }, { x: 10, y: 10 }, { x:
11, y: 16 }, { x: 12, y: 6 },
    { x: 13, y: 14 }, { x: 14, y: 7 }, { x: 15, y: 5 }, { x: 16, y: 2 }, {
x: 17, y: 14 }, { x: 18, y: 7 },
    { x: 19, y: 7 }, { x: 20, y: 10 }];
var chart3D = new ej.charts.Chart3D({
    primaryXAxis: {
        valueType: 'Double',
        minimum: 1,
        maximum: 20,
        interval: 5
    },
    series: [
        {
            type: 'Column', xName: 'x', yName: 'y', columnSpacing: 0.1,
            dataSource: chartData
        }
    ],
    wallColor: 'transparent',
    enableRotation: true,
    rotation: 7,
    tilt: 10,
    depth: 100
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">

```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Range padding

Padding can be applied to the minimum and maximum extremes of an axis range by using the [rangePadding](#) property. Numeric axis supports the following types of padding.

- None
- Round
- Additional
- Normal
- Auto

Numeric - None

When the [rangePadding](#) is set to **None**, minimum and maximum of the axis is based on the data.

INDEX.JS

```

var chartData = [{ x: 1, y: 7 }, { x: 2, y: 1 }, { x: 3, y: 1 }, { x: 4, y:
14 }, { x: 5, y: 1 }, { x: 6, y: 10 },
    { x: 7, y: 8 }, { x: 8, y: 6 }, { x: 9, y: 10 }, { x: 10, y: 10 }, { x:
11, y: 16 }, { x: 12, y: 6 },
    { x: 13, y: 14 }, { x: 14, y: 7 }, { x: 15, y: 5 }, { x: 16, y: 2 }, {
x: 17, y: 14 }, { x: 18, y: 7 },
    { x: 19, y: 7 }, { x: 20, y: 10 }];
var chart3D = new ej.charts.Chart3D({
    primaryXAxis: {
        valueType: 'Double'
    },
    primaryYAxis: {
        //RangePadding as none in Y Axis

```

```

        rangePadding: 'None'
    },
    series: [
        {
            type: 'Column', xName: 'x', yName: 'y', columnSpacing: 0.1,
            dataSource: chartData
        }
    ],
    wallColor: 'transparent',
    enableRotation: true,
    rotation: 7,
    tilt: 10,
    depth: 100
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Numeric - Round

When the [rangePadding](#) is set to **Round**, minimum and maximum will be rounded to the nearest possible value, which is divisible by interval. For example, when the [minimum](#) is 3.5 and the [interval](#) is 1, then the minimum will be rounded to 3.

INDEX.JS

```

var chartData = [{ x: 1, y: 7 }, { x: 2, y: 1 }, { x: 3, y: 1 }, { x: 4, y:
14 }, { x: 5, y: 1 }, { x: 6, y: 10 },

```

```

    { x: 7, y: 8 }, { x: 8, y: 6 }, { x: 9, y: 10 }, { x: 10, y: 10 }, { x:
11, y: 16 }, { x: 12, y: 6 },
    { x: 13, y: 14 }, { x: 14, y: 7 }, { x: 15, y: 5 }, { x: 16, y: 2 }, {
x: 17, y: 14 }, { x: 18, y: 7 },
    { x: 19, y: 7 }, { x: 20, y: 10 }];
var chart3D = new ej.charts.Chart3D({
    primaryXAxis: {
        valueType: 'Double'
    },
    primaryYAxis: {
        rangePadding: 'Round'
    },
    series: [
        {
            type: 'Column', xName: 'x', yName: 'y', columnSpacing: 0.1,
            dataSource: chartData
        }
    ],
    wallColor: 'transparent',
    enableRotation: true,
    rotation: 7,
    tilt: 10,
    depth: 100
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

Numeric - Additional

When the [rangePadding](#) is set to **Additional**, interval of an axis will be added to the minimum and maximum of the axis.

INDEX.JS

```
var chartData = [{ x: 1, y: 7 }, { x: 2, y: 1 }, { x: 3, y: 1 }, { x: 4, y: 14 }, { x: 5, y: 1 }, { x: 6, y: 10 }, { x: 7, y: 8 }, { x: 8, y: 6 }, { x: 9, y: 10 }, { x: 10, y: 10 }, { x: 11, y: 16 }, { x: 12, y: 6 }, { x: 13, y: 14 }, { x: 14, y: 7 }, { x: 15, y: 5 }, { x: 16, y: 2 }, { x: 17, y: 14 }, { x: 18, y: 7 }, { x: 19, y: 7 }, { x: 20, y: 10 }];
var chart3D = new ej.charts.Chart3D({
  primaryXAxis: {
    valueType: 'Double'
  },
  primaryYAxis: {
    rangePadding: 'Additional'
  },
  series: [
    {
      type: 'Column', xName: 'x', yName: 'y', columnSpacing: 0.1,
      dataSource: chartData
    }
  ],
  wallColor: 'transparent',
  enableRotation: true,
  rotation: 7,
  tilt: 10,
  depth: 100
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
```

```

if(ele) {
    ele.style.visibility = "visible";
}

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Numeric - Normal

When the [rangePadding](#) is set to **Normal**, padding is applied to the axis based on default range calculation.

INDEX.JS

```

var chartData = [{ x: 1, y: 7 }, { x: 2, y: 1 }, { x: 3, y: 1 }, { x: 4, y: 14 }, { x: 5, y: 1 }, { x: 6, y: 10 }, { x: 7, y: 8 }, { x: 8, y: 6 }, { x: 9, y: 10 }, { x: 10, y: 10 }, { x: 11, y: 16 }, { x: 12, y: 6 }, { x: 13, y: 14 }, { x: 14, y: 7 }, { x: 15, y: 5 }, { x: 16, y: 2 }, { x: 17, y: 14 }, { x: 18, y: 7 }, { x: 19, y: 7 }, { x: 20, y: 10 }];
var chart3D = new ej.charts.Chart3D({
    primaryXAxis: {
        valueType: 'Double'
    },
    primaryYAxis: {
        rangePadding: 'Normal'
    },
    series: [
        {
            type: 'Column', xName: 'x', yName: 'y', columnSpacing: 0.1,
            dataSource: chartData
        }
    ],
    wallColor: 'transparent',
    enableRotation: true,
    rotation: 7,
    tilt: 10,
    depth: 100
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>

```

```

</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Numeric - Auto

When the [rangePadding](#) is set to **Auto**, horizontal numeric axis takes **None** as padding calculation, while the vertical numeric axis takes **Normal** as padding calculation.

INDEX.JS

```

var chartData = [{ x: 1, y: 7 }, { x: 2, y: 1 }, { x: 3, y: 1 }, { x: 4, y: 14 }, { x: 5, y: 1 }, { x: 6, y: 10 }, { x: 7, y: 8 }, { x: 8, y: 6 }, { x: 9, y: 10 }, { x: 10, y: 10 }, { x: 11, y: 16 }, { x: 12, y: 6 }, { x: 13, y: 14 }, { x: 14, y: 7 }, { x: 15, y: 5 }, { x: 16, y: 2 }, { x: 17, y: 14 }, { x: 18, y: 7 }, { x: 19, y: 7 }, { x: 20, y: 10 }];
var chart3D = new ej.charts.Chart3D({
    primaryXAxis: {
        valueType: 'Double',
        // Set the rangePadding as auto in X Axis
        rangePadding: 'Auto'
    },
    primaryYAxis: {
        rangePadding: 'Auto'
    },
    series: [
        {
            type: 'Column', xName: 'x', yName: 'y', columnSpacing: 0.1,
            dataSource: chartData
        }
    ],
    wallColor: 'transparent',
    enableRotation: true,
    rotation: 7,
    tilt: 10,
    depth: 100
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">

```



```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Label format

Numeric label format

Numeric labels can be formatted by using the [labelFormat](#) property. Also, it supports all globalize format.

INDEX.JS

```

var chartData = [{ x: 1, y: 7 }, { x: 2, y: 1 }, { x: 3, y: 1 }, { x: 4, y:
14 }, { x: 5, y: 1 }, { x: 6, y: 10 },
    { x: 7, y: 8 }, { x: 8, y: 6 }, { x: 9, y: 10 }, { x: 10, y: 10 }, { x:
11, y: 16 }, { x: 12, y: 6 },
    { x: 13, y: 14 }, { x: 14, y: 7 }, { x: 15, y: 5 }, { x: 16, y: 2 }, {
x: 17, y: 14 }, { x: 18, y: 7 },
    { x: 19, y: 7 }, { x: 20, y: 10 }];
var chart3D = new ej.charts.Chart3D({
    primaryXAxis: {
        valueType: 'Double'
    },
    primaryYAxis: {
        labelFormat: 'c'
    },
    series: [
        {
            type: 'Column', xName: 'x', yName: 'y', columnSpacing: 0.1,
            dataSource: chartData
        }
    ],
    wallColor: 'transparent',
    enableRotation: true,
    rotation: 7,

```

```

    tilt: 10,
    depth: 100
  }, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

The following table describes the result of applying some commonly used label formats on numeric values.

<!-- markdownlint-disable MD033 -->

Label Value	Label Format: Property Value	Result	Description
1000	n1	1000.0	The Number is rounded to 1 decimal place.
1000	n2	1000.00	The Number is rounded to 2 decimal place.
1000	n3	1000.000	The Number is rounded to 3 decimal place.
0.01	p1	1.0%	The Number is converted to percentage with 1 decimal place.
0.01	p2	1.00%	The Number is converted to percentage with 2 decimal place.
0.01	p3	1.000%	The Number is converted to percentage with 3 decimal place.

1000	c1	\$1000.0	The Currency symbol is appended to number and number is rounded to 1 decimal place.
1000	c2	\$1000.00	The Currency symbol is appended to number and number is rounded to 2 decimal place.

Grouping separator

To separate the y-axis labels to groups of thousands, set the [useGroupingSeparator](#) property to **true** in the 3D chart.

INDEX.JS

```
var chartData = [
  { x: 10, y: 7000 }, { x: 20, y: 1000 }, { x: 30, y: 12000 }, { x: 40, y:
  14000 }, { x: 50, y: 11000 }, { x: 60, y: 5000 },
  { x: 70, y: 7300 }, { x: 80, y: 9000 }, { x: 90, y: 12000 }, { x: 100,
  y: 14000 }, { x: 110, y: 11000 }, { x: 120, y: 5000 }
];
var chart3D = new ej.charts.Chart3D({
  useGroupingSeparator: true,
  series: [
    {
      type: 'Column', xName: 'x', yName: 'y', columnSpacing: 0.1,
      dataSource: chartData
    }
  ],
  wallColor: 'transparent',
  enableRotation: true,
  rotation: 7,
  tilt: 10,
  depth: 100
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
```

```
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Custom label format

The axis supports custom label format using placeholder like **{value}°C**, in which the value represent the axis label e.g 20°C.

INDEX.JS

```
var chartData = [
    { x: 1, y: 7 }, { x: 2, y: 1 }, { x: 3, y: 1 }, { x: 4, y: 14 }, { x: 5,
y: 1 }, { x: 6, y: 10 },
    { x: 7, y: 8 }, { x: 8, y: 6 }, { x: 9, y: 10 }, { x: 10, y: 10 }, { x:
11, y: 16 }, { x: 12, y: 6 },
    { x: 13, y: 14 }, { x: 14, y: 7 }, { x: 15, y: 5 }, { x: 16, y: 2 }, {
x: 17, y: 14 }, { x: 18, y: 7 },
    { x: 19, y: 7 }, { x: 20, y: 10 }
];
var chart3D = new ej.charts.Chart3D({
    primaryYAxis: {
        // Custom label format
        labelFormat: '${value}K'
    },
    series: [
        {
            type: 'Column', xName: 'x', yName: 'y', columnSpacing: 0.1,
            dataSource: chartData
        }
    ],
    wallColor: 'transparent',
    enableRotation: true,
    rotation: 7,
    tilt: 10,
    depth: 100
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

<!-- markdownlint-disable MD036 -->

DateTime axis in EJ2 JavaScript 3D Chart control

DateTime axis

DateTime axis uses date time scale and displays the date time values as axis labels in the specified format.

INDEX.JS

```

var chartData = [
    { x: new Date(2000, 6, 11), y: 10 },
    { x: new Date(2002, 3, 7), y: 30 },
    { x: new Date(2004, 3, 6), y: 15 },
    { x: new Date(2006, 3, 30), y: 65 },
    { x: new Date(2008, 3, 8), y: 90 },
    { x: new Date(2010, 3, 8), y: 85 }
];
var chart3D = new ej.charts.Chart3D({
    primaryXAxis: {
        // Date time scale in primary X Axis
        valueType: 'DateTime',
    },
    series:[{
        dataSource: chartData,
        xName: 'x', yName: 'y',
        type: 'Column'
    }],
    wallColor: 'transparent',
    enableRotation: true,
    rotation: 7,
    tilt: 10,
    depth: 100
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>

```

```

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Note: To use datetime axis, we need to inject `DateTime3D` module using `Chart3D.Inject(DateTime3D)` method and set the [valueType](#) of axis to `DateTime`.

DateTime category axis

DateTime category axis is used to display the date time values with non-linear intervals. For example, the business days alone have been depicted in a week here.

INDEX.JS

```

var chartData = [
    { x: new Date(2017, 11, 20), y: 21 }, { x: new Date(2017, 11, 21), y: 24 },
    { x: new Date(2017, 11, 22), y: 24 }, { x: new Date(2017, 11, 26), y: 70 },
    { x: new Date(2017, 11, 27), y: 75 }, { x: new Date(2018, 0, 2), y: 82 },
    { x: new Date(2018, 0, 3), y: 53 }, { x: new Date(2018, 0, 4), y: 54 },
    { x: new Date(2018, 0, 5), y: 53 }, { x: new Date(2018, 0, 8), y: 45 }
];
var chart3D = new ej.charts.Chart3D({
    primaryXAxis: {
        valueType: 'DateTimeCategory',
        skeleton: 'Ed',
    },
    series: [
        {
            type: 'Column', xName: 'x', yName: 'y',
            dataSource: chartData
        }
    ]
});

```

```

    wallColor: 'transparent',
    enableRotation: true,
    rotation: 7,
    tilt: 10,
    depth: 100
  }, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Note: To use datetime category axis, we need to inject `DateTimeCategory3D` module using the `Chart3D.Inject(DateTimeCategory3D)` method and set the [valueType](#) of axis to **DateTimeCategory**.

Range

Range of an axis will be calculated automatically based on the provided data. You can also customize the range of an axis using [minimum](#), [maximum](#) and [interval](#) properties.

INDEX.JS

```

var chartData = [
  { x: new Date(2000, 6, 11), y: 10 }, { x: new Date(2002, 3, 7), y: 30 },
  { x: new Date(2004, 3, 6), y: 15 }, { x: new Date(2006, 3, 30), y: 65 },
  { x: new Date(2008, 3, 8), y: 90 }, { x: new Date(2010, 3, 8), y: 85 }
];
var chart3D = new ej.charts.Chart3D({
  primaryXAxis: {
    valueType: 'DateTime',
    minimum: new Date(2000, 6, 1),

```

```

        maximum: new Date(2010, 6, 1), interval: 1
    },
    series: [
        {
            type: 'Column', xName: 'x', yName: 'y',
            dataSource: chartData
        }
    ],
    wallColor: 'transparent',
    enableRotation: true,
    rotation: 7,
    tilt: 10,
    depth: 100
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Interval customization

Date time intervals can be customized by using the [interval](#) and [intervalType](#) properties of the axis. For example, when you set [interval](#) as **2** and [intervalType](#) as **Years**, it considers 2 years as interval.

DateTime axis supports following interval types,

- Auto
- Years
- Months
- Days

- Hours
- Minutes
- Seconds

INDEX.JS

```

var chartData = [
  { x: new Date(2000, 6, 11), y: 10 }, { x: new Date(2002, 3, 7), y: 30 },
  { x: new Date(2004, 3, 6), y: 15 }, { x: new Date(2006, 3, 30), y: 65 },
  { x: new Date(2008, 3, 8), y: 90 }, { x: new Date(2010, 3, 8), y: 85 }
];
var chart3D = new ej.charts.Chart3D({
  primaryXAxis: {
    valueType: 'DateTime',
    intervalType: 'Years'
  },
  series: [
    {
      type: 'Column', xName: 'x', yName: 'y',
      dataSource: chartData
    }
  ],
  wallColor: 'transparent',
  enableRotation: true,
  rotation: 7,
  tilt: 10,
  depth: 100
}, '#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
```

```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Applying padding to the range

Padding can be applied to the minimum and maximum extremes of the range by using the [rangePadding](#) property. DateTime axis supports the following types of padding,

- None
- Round
- Additional

DateTime - None

When the [rangePadding](#) is set to **None**, minimum and maximum of an axis is based on the data.

INDEX.JS

```
var chartData = [
  { x: new Date(2017, 11, 20), y: 21 }, { x: new Date(2017, 11, 21), y: 24 },
  { x: new Date(2017, 11, 22), y: 24 }, { x: new Date(2017, 11, 26), y: 70 },
  { x: new Date(2017, 11, 27), y: 75 }, { x: new Date(2018, 0, 2), y: 82 },
  { x: new Date(2018, 0, 3), y: 53 }, { x: new Date(2018, 0, 4), y: 54 },
  { x: new Date(2018, 0, 5), y: 53 }, { x: new Date(2018, 0, 8), y: 45 }
];
var chart3D = new ej.charts.Chart3D({
  primaryXAxis: {
    valueType: 'DateTime',
    rangePadding: 'None'
  },
  series: [
    {
      type: 'Column', xName: 'x', yName: 'y',
      dataSource: chartData
    }
  ],
  wallColor: 'transparent',
  enableRotation: true,
  rotation: 7,
  tilt: 10,
  depth: 100
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

DateTime - Round

When the [rangePadding](#) is set to **Round**, minimum and maximum will be rounded to the nearest possible value, which is divisible by interval. For example, when the minimum is **15th Jan**, interval is **1** and interval type is **Month**, then the axis minimum will be **Jan 1st**.

INDEX.JS

```

var chartData = [
    { x: new Date(2017, 11, 20), y: 21 }, { x: new Date(2017, 11, 21), y: 24 },
    { x: new Date(2017, 11, 22), y: 24 }, { x: new Date(2017, 11, 26), y: 70 },
    { x: new Date(2017, 11, 27), y: 75 }, { x: new Date(2018, 0, 2), y: 82 },
    { x: new Date(2018, 0, 3), y: 53 }, { x: new Date(2018, 0, 4), y: 54 },
    { x: new Date(2018, 0, 5), y: 53 }, { x: new Date(2018, 0, 8), y: 45 }
];
var chart3D = new ej.charts.Chart3D({
    primaryXAxis: {
        valueType: 'DateTime',
        rangePadding: 'Round'
    },
    series: [
        {
            type: 'Column', xName: 'x', yName: 'y',
            dataSource: chartData
        }
    ],
    wallColor: 'transparent',
    enableRotation: true,
    rotation: 7,
    tilt: 10,
    depth: 100
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

DateTime - Additional

When the [rangePadding](#) is set to **Additional**, interval of an axis will be padded to the minimum and maximum of the axis.

INDEX.JS

```

var chartData = [
  { x: new Date(2017, 11, 20), y: 21 }, { x: new Date(2017, 11, 21), y: 24 },
],
  { x: new Date(2017, 11, 22), y: 24 }, { x: new Date(2017, 11, 26), y: 70 },
],
  { x: new Date(2017, 11, 27), y: 75 }, { x: new Date(2018, 0, 2), y: 82 },
],
  { x: new Date(2018, 0, 3), y: 53 }, { x: new Date(2018, 0, 4), y: 54 },
  { x: new Date(2018, 0, 5), y: 53 }, { x: new Date(2018, 0, 8), y: 45 }
];
var chart3D = new ej.charts.Chart3D({
  primaryXAxis: {
    valueType: 'DateTime',
    rangePadding: 'Additional'
  },
  series: [
    {
      type: 'Column', xName: 'x', yName: 'y',

```

```

        dataSource: chartData
    }
    ],
    wallColor: 'transparent',
    enableRotation: true,
    rotation: 7,
    tilt: 10,
    depth: 100
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Label format

The date can be formatted and parsed to all globalize format using the [labelFormat](#) property in an axis.

INDEX.JS

```

var chartData = [
  { x: new Date(2017, 11, 20), y: 21 }, { x: new Date(2017, 11, 21), y: 24
},
  { x: new Date(2017, 11, 22), y: 24 }, { x: new Date(2017, 11, 26), y: 70
},
  { x: new Date(2017, 11, 27), y: 75 }, { x: new Date(2018, 0, 2), y: 82
},
  { x: new Date(2018, 0, 3), y: 53 }, { x: new Date(2018, 0, 4), y: 54 },
  { x: new Date(2018, 0, 5), y: 53 }, { x: new Date(2018, 0, 8), y: 45 }
];
var chart3D = new ej.charts.Chart3D({

```

```

primaryXAxis: {
    valueType: 'DateTime',
    labelFormat: 'yMd'
},
series: [
    {
        type: 'Column', xName: 'x', yName: 'y',
        dataSource: chartData
    }
],
wallColor: 'transparent',
enableRotation: true,
rotation: 7,
tilt: 10,
depth: 100
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

The following table describes the result of applying some common date time formats to the **labelFormat** property.

<!-- markdownlint-disable MD033 -->

Label Value	Label Format Property Value	Result	Description
new Date(2000, 03, 10)	EEEE	Monday	The Date is displayed in day format.

new Date(2000, 03, 10)	yMd	04/10/2000	The Date is displayed in month/date/year format.
new Date(2000, 03, 10)	MMM	Apr	The Shorthand month for the date is displayed.
new Date(2000, 03, 10)	hm	12:00 AM	Time of the date value is displayed as label.
new Date(2000, 03, 10)	hms	12:00:00 AM	The Label is displayed in hours:minutes:seconds format.

Custom label format

Axis also supports custom label format using placeholder like {value}°C, in which the value represent the axis label e.g 20°C.

INDEX.JS

```
var chartData = [
  { x: new Date(2017, 11, 20), y: 21 }, { x: new Date(2017, 11, 21), y: 24 },
  { x: new Date(2017, 11, 22), y: 24 }, { x: new Date(2017, 11, 26), y: 70 },
  { x: new Date(2017, 11, 27), y: 75 }, { x: new Date(2018, 0, 2), y: 82 },
  { x: new Date(2018, 0, 3), y: 53 }, { x: new Date(2018, 0, 4), y: 54 },
  { x: new Date(2018, 0, 5), y: 53 }, { x: new Date(2018, 0, 8), y: 45 }
];
var chart3D = new ej.charts.Chart3D({
  primaryXAxis: {
    valueType: 'DateTimeCategory'
  },
  primaryYAxis: {
    labelFormat: '${value}K'
  },
  series: [
    {
      type: 'Column', xName: 'x', yName: 'y',
      dataSource: chartData
    }
  ],
  wallColor: 'transparent',
  enableRotation: true,
  rotation: 7,
  tilt: 10,
  depth: 100
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```

<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Logarithmic axis in EJ2 JavaScript 3D Chart control

Logarithmic axis uses logarithmic scale and it is very useful in visualizing data, when it has numerical values in both lower order of magnitude (eg: 10^{-6}) and higher order of magnitude (eg: 10^6).

INDEX.JS

```

var chartData = [
    { x: new Date(1995, 0, 1), y: 80 }, { x: new Date(1996, 0, 1), y: 200 },
    { x: new Date(1997, 0, 1), y: 400 }, { x: new Date(1998, 0, 1), y: 600
},
    { x: new Date(1999, 0, 1), y: 700 }, { x: new Date(2000, 0, 1), y: 1400
},
    { x: new Date(2001, 0, 1), y: 2000 }, { x: new Date(2002, 0, 1), y: 4000
},
    { x: new Date(2003, 0, 1), y: 6000 }, { x: new Date(2004, 0, 1), y: 8000
},
    { x: new Date(2005, 0, 1), y: 11000 }
];
var chart3D = new ej.charts.Chart3D({
    primaryXAxis: {
        valueType: 'DateTime'
    },
    primaryYAxis:
    {
        valueType: 'Logarithmic'
    },
    series: [
        {
            type: 'Column', xName: 'x', yName: 'y',
            dataSource: chartData
        }
    ]
}

```



```

    ],
    wallColor: 'transparent',
    enableRotation: true,
    rotation: 7,
    tilt: 10,
    depth: 100
  }, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Note: To use logarithmic axis, we need to inject **Logarithmic3D** module using the **Chart3D.Inject(Logarithmic3D)** method and set the [valueType](#) of the axis to **Logarithmic**.

Range

The range of an axis will be calculated automatically based on the provided data and it can also be customized by using the [minimum](#), [maximum](#) and [interval](#) properties of the axis.

INDEX.JS

```

var chartData = [
  { x: new Date(1995, 0, 1), y: 80 }, { x: new Date(1996, 0, 1), y: 200 },
  { x: new Date(1997, 0, 1), y: 400 }, { x: new Date(1998, 0, 1), y: 600
},
  { x: new Date(1999, 0, 1), y: 700 }, { x: new Date(2000, 0, 1), y: 1400
},
  { x: new Date(2001, 0, 1), y: 2000 }, { x: new Date(2002, 0, 1), y: 4000
},

```

```

    { x: new Date(2003, 0, 1), y: 6000 }, { x: new Date(2004, 0, 1), y: 8000
  },
  { x: new Date(2005, 0, 1), y: 11000 }
];
var chart3D = new ej.charts.Chart3D({
  primaryXAxis: {
    valueType: 'DateTime'
  },
  primaryYAxis:
  {
    valueType: 'Logarithmic',
    minimum: 100,
    maximum: 10000,
    interval: 1000
  },
  series: [
    {
      type: 'Column', xName: 'x', yName: 'y',
      dataSource: chartData
    }
  ],
  wallColor: 'transparent',
  enableRotation: true,
  rotation: 7,
  tilt: 10,
  depth: 100
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
<script src="index.js" type="text/javascript"></script>

```

```
</body></html>
```

Logarithmic base

Logarithmic base can be customized by using the [logBase](#) property of the axis. For example when the **logBase** is 5, the axis values follows 5^{-2} , 5^{-1} , 5^0 , 5^1 , 5^2 etc.

INDEX.JS

```
var chartData = [
  { x: new Date(1995, 0, 1), y: 80 }, { x: new Date(1996, 0, 1), y: 200 },
  { x: new Date(1997, 0, 1), y: 400 }, { x: new Date(1998, 0, 1), y: 600 },
  { x: new Date(1999, 0, 1), y: 700 }, { x: new Date(2000, 0, 1), y: 1400 },
  { x: new Date(2001, 0, 1), y: 2000 }, { x: new Date(2002, 0, 1), y: 4000 },
  { x: new Date(2003, 0, 1), y: 6000 }, { x: new Date(2004, 0, 1), y: 8000 },
  { x: new Date(2005, 0, 1), y: 11000 }
];
var chart3D = new ej.charts.Chart3D({
  primaryXAxis: {
    valueType: 'DateTime'
  },
  primaryYAxis: {
    valueType: 'Logarithmic',
    logBase: 2
  },
  //Initializing Chart Series
  series: [
    {
      type: 'Column', xName: 'x', yName: 'y', columnSpacing: 0.1,
      dataSource: chartData
    }
  ],
  wallColor: 'transparent',
  enableRotation: true,
  rotation: 7,
  tilt: 10,
  depth: 100
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Logarithmic interval

The interval of the logarithmic axis can be customized by using the [interval](#) property in the axis. When the logarithmic base is 10 and logarithmic **interval** is 2, then the axis labels are placed at an interval of 10^2 . The default value of the [interval](#) is 1.

INDEX.JS

```

var chartData = [
    { x: new Date(1995, 0, 1), y: 80 }, { x: new Date(1996, 0, 1), y: 200 },
    { x: new Date(1997, 0, 1), y: 400 }, { x: new Date(1998, 0, 1), y: 600 },
    { x: new Date(1999, 0, 1), y: 700 }, { x: new Date(2000, 0, 1), y: 1400 },
    { x: new Date(2001, 0, 1), y: 2000 }, { x: new Date(2002, 0, 1), y: 4000 },
    { x: new Date(2003, 0, 1), y: 6000 }, { x: new Date(2004, 0, 1), y: 8000 },
    { x: new Date(2005, 0, 1), y: 11000 }
];
var chart3D = new ej.charts.Chart3D({
    primaryXAxis: {
        valueType: 'DateTime'
    },
    primaryYAxis: {
        valueType: 'Logarithmic',
        interval: 2
    },
    series: [
        {
            type: 'Column', xName: 'x', yName: 'y',
            dataSource: chartData
        }
    ],
    wallColor: 'transparent',
    enableRotation: true,
    rotation: 7,

```

```

    tilt: 10,
    depth: 100
  }, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Axis labels in EJ2 JavaScript 3D Chart control

Axis labels are the labels that are positioned adjacent to the y-axis and beneath the x-axis. It provides descriptive information about the axis.

Smart axis labels

When the axis labels overlap with each other, [labelIntersectAction](#) property in the axis can be used to place them smartly.

Case 1: When setting `labelIntersectAction` as `Hide`.

INDEX.JS

```

var chartData = [
  { x: "South Korea", y: 39.4 },
  { x: "India", y: 61.3 },
  { x: "Pakistan", y: 20.4 },
  { x: "Germany", y: 65.1 },
  { x: "Australia", y: 15.8 },
  { x: "Italy", y: 29.2 },
  { x: "United Kingdom", y: 44.6 },
  { x: "Saudi Arabia", y: 9.7 },

```

```

    { x: "Russia", y: 40.8 },
    { x: "Mexico", y: 31 },
    { x: "Brazil", y: 75.9 },
    { x: "China", y: 51.4 }
  ];
  var chart3D = new ej.charts.Chart3D({
    primaryXAxis: {
      valueType: 'Category',
      labelIntersectAction: 'Hide'
    },
    series: [
      {
        type: 'Column', xName: 'x', yName: 'y',
        dataSource: chartData
      }
    ],
    wallColor: 'transparent',
    enableRotation: true,
    rotation: 7,
    tilt: 10,
    depth: 100
  }, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
  var ele = document.getElementById('container');
  if(ele) {
    ele.style.visibility = "visible";
  }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Case 2: When setting `labelIntersectAction` as `Rotate45`.

INDEX.JS

```

var chartData = [
  { x: "South Korea", y: 39.4 },
  { x: "India", y: 61.3 },
  { x: "Pakistan", y: 20.4 },
  { x: "Germany", y: 65.1 },
  { x: "Australia", y: 15.8 },
  { x: "Italy", y: 29.2 },
  { x: "United Kingdom", y: 44.6 },
  { x: "Saudi Arabia", y: 9.7 },
  { x: "Russia", y: 40.8 },
  { x: "Mexico", y: 31 },
  { x: "Brazil", y: 75.9 },
  { x: "China", y: 51.4 }
];
var chart3D = new ej.charts.Chart3D({
  primaryXAxis: {
    valueType: 'Category',
    labelIntersectAction: 'Rotate45'
  },
  series: [
    {
      type: 'Column', xName: 'x', yName: 'y',
      dataSource: chartData
    }
  ],
  wallColor: 'transparent',
  enableRotation: true,
  rotation: 7,
  tilt: 10,
  depth: 100
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
</body>
</script>

```

```

var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Case 3: When setting `labelIntersectAction` as `Rotate90`.

INDEX.JS

```

var chartData = [
    { x: "South Korea", y: 39.4 },
    { x: "India", y: 61.3 },
    { x: "Pakistan", y: 20.4 },
    { x: "Germany", y: 65.1 },
    { x: "Australia", y: 15.8 },
    { x: "Italy", y: 29.2 },
    { x: "United Kingdom", y: 44.6 },
    { x: "Saudi Arabia", y: 9.7 },
    { x: "Russia", y: 40.8 },
    { x: "Mexico", y: 31 },
    { x: "Brazil", y: 75.9 },
    { x: "China", y: 51.4 }
];
var chart3D = new ej.charts.Chart3D({
    primaryXAxis: {
        valueType: 'Category',
        labelIntersectAction: 'Rotate90'
    },
    series: [
        {
            type: 'Column', xName: 'x', yName: 'y',
            dataSource: chartData
        }
    ],
    wallColor: 'transparent',
    enableRotation: true,
    rotation: 7,
    tilt: 10,
    depth: 100
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

```



```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Edge label placement

Labels with long text at the edges of an axis may appear partially in the 3D chart. To avoid this, use the [edgeLabelPlacement](#) property in axis, which moves the label inside the chart area for better appearance or hides it.

INDEX.JS

```

var chartData = [
    { country: "USA", gold: 50, silver: 70, bronze: 45 },
    { country: "China", gold: 40, silver: 60, bronze: 55 },
    { country: "Japan", gold: 70, silver: 60, bronze: 50 },
    { country: "Australia", gold: 60, silver: 56, bronze: 40 },
    { country: "France", gold: 50, silver: 45, bronze: 35 },
    { country: "Germany", gold: 40, silver: 30, bronze: 22 },
    { country: "Italy", gold: 40, silver: 35, bronze: 37 },
    { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart3D = new ej.charts.Chart3D({
    primaryXAxis: {
        valueType: 'Category',
        edgeLabelPlacement: 'Shift',
    },
    series: [
        {
            type: 'Column', xName: 'country', yName: 'gold',
            dataSource: chartData
        }
    ],
    wallColor: 'transparent',
    enableRotation: true,
    rotation: 7,
    tilt: 10,
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Maximum labels

The labels will be rendered based on the count in the [maximumLabels](#) property per 100 pixel. If the range (minimum, maximum, interval) and [maximumLabels](#) are set, then the priority goes to range. If the range is not set, then the priority goes to [maximumLabels](#) property.

INDEX.JS

```

var series1 = [];
var point1;
var value = 80;
for (var i = 1; i < 50; i++) {
  if (Math.random() > .5) {
    value += Math.random();
  } else {
    value -= Math.random();
  }
  point1 = { x: i, y: value.toFixed(1) };
  series1.push(point1);
}
var chart3D = new ej.charts.Chart3D({
  primaryXAxis: {
    title: 'Years',
    edgeLabelPlacement: 'Shift',
    majorGridLines: { width: 0 },
    maximumLabels: 1,
  },
  //Initializing Primary Y Axis

```

```

primaryYAxis:
{
    title: 'Profit ($)',
    rangePadding: 'None',
    majorTickLines: { width: 0 }
},
series: [
    {
        type: 'Column',
        dataSource: series1,
        name: 'Product X',
        xName: 'x',
        yName: 'y',
        animation: { enable: false }
    },
],
title: 'Sales History of Product X',
wallColor: 'transparent',
enableRotation: true,
rotation: 7,
tilt: 10,
depth: 100
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Axis customization in EJ2 JavaScript 3D Chart control

Title

The title for the axis can be added by using the [title](#) property. It helps to provide quick information to the user about the data plotted in the axis. Title style can be customized using [titleStyle](#) property of the axis.

INDEX.JS

```
var chartData = [
  { country: "USA", gold: 50, silver: 70, bronze: 45 },
  { country: "China", gold: 40, silver: 60, bronze: 55 },
  { country: "Japan", gold: 70, silver: 60, bronze: 50 },
  { country: "Australia", gold: 60, silver: 56, bronze: 40 },
  { country: "France", gold: 50, silver: 45, bronze: 35 },
  { country: "Germany", gold: 40, silver: 30, bronze: 22 },
  { country: "Italy", gold: 40, silver: 35, bronze: 37 },
  { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart3D = new ej.charts.Chart3D({
  primaryXAxis: {
    valueType: "Category",
    title: 'Countries',
    //Axis title text style
    titleStyle: {
      size: '16px', color: 'grey',
      fontFamily : 'Segoe UI', fontWeight : 'bold'
    }
  },
  series: [
    {
      type: 'Column',
      dataSource: chartData,
      xName: 'country',
      yName: 'gold'
    }
  ],
  wallColor: 'transparent',
  enableRotation: true,
  rotation: 7,
  tilt: 10,
  depth: 100
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Title rotation

The title can be rotated from 0 to 360 degree by using the [titleRotation](#) property.

INDEX.JS

```

var chartData = [
    { country: "USA", gold: 50, silver: 70, bronze: 45 },
    { country: "China", gold: 40, silver: 60, bronze: 55 },
    { country: "Japan", gold: 70, silver: 60, bronze: 50 },
    { country: "Australia", gold: 60, silver: 56, bronze: 40 },
    { country: "France", gold: 50, silver: 45, bronze: 35 },
    { country: "Germany", gold: 40, silver: 30, bronze: 22 },
    { country: "Italy", gold: 40, silver: 35, bronze: 37 },
    { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart3D = new ej.charts.Chart3D({
    primaryXAxis: {
        valueType: "Category",
        title: 'Countries',
        titleRotation: 90,
        //Axis title text style
        titleStyle: {
            size: '16px', color: 'grey',
            fontFamily : 'Segoe UI', fontWeight : 'bold'
        }
    },
    series: [
        {
            type: 'Column',
            dataSource: chartData,
            xName: 'country',
            yName: 'gold'
        },
    ],
    wallColor: 'transparent',
    enableRotation: true,
    rotation: 7,

```

```

    tilt: 10,
    depth: 100
  }, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Tick lines customization

The [width](#), [color](#) and [height](#) of the minor and major tick lines can be customized by using the [majorTickLines](#) and [minorTickLines](#) properties in the axis.

INDEX.JS

```

var chartData = [
  { country: "USA", gold: 50, silver: 70, bronze: 45 },
  { country: "China", gold: 40, silver: 60, bronze: 55 },
  { country: "Japan", gold: 70, silver: 60, bronze: 50 },
  { country: "Australia", gold: 60, silver: 56, bronze: 40 },
  { country: "France", gold: 50, silver: 45, bronze: 35 },
  { country: "Germany", gold: 40, silver: 30, bronze: 22 },
  { country: "Italy", gold: 40, silver: 35, bronze: 37 },
  { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart3D = new ej.charts.Chart3D({
  primaryXAxis: {
    valueType: "Category",
    majorTickLines : {
      color : 'blue',

```

```

        width : 5
    },
    series: [
        {
            type: 'Column',
            dataSource: chartData,
            xName: 'country',
            yName: 'gold'
        },
    ],
    wallColor: 'transparent',
    enableRotation: true,
    rotation: 7,
    tilt: 10,
    depth: 100
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

Grid lines customization

The [width](#) and [color](#) of the minor and major grid lines can be customized by using the [majorGridLines](#) and [minorGridLines](#) properties in the axis.

INDEX.JS

```

var chartData = [

```

```

    { country: "USA", gold: 50, silver: 70, bronze: 45 },
    { country: "China", gold: 40, silver: 60, bronze: 55 },
    { country: "Japan", gold: 70, silver: 60, bronze: 50 },
    { country: "Australia", gold: 60, silver: 56, bronze: 40 },
    { country: "France", gold: 50, silver: 45, bronze: 35 },
    { country: "Germany", gold: 40, silver: 30, bronze: 22 },
    { country: "Italy", gold: 40, silver: 35, bronze: 37 },
    { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
  ];
  var chart3D = new ej.charts.Chart3D({
    primaryXAxis: {
      valueType: "Category",
      majorGridLines : {
        color : 'blue',
        width : 1
      }
    },
    series: [
      {
        type: 'Column',
        dataSource: chartData,
        xName: 'country',
        yName: 'gold'
      },
    ],
    wallColor: 'transparent',
    enableRotation: true,
    rotation: 7,
    tilt: 10,
    depth: 100
  }, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {

```



```

    ele.style.visibility = "visible";
  }
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Multiple axis

In addition to primary X and Y axis, n number of axis can be added to the chart. Series can be associated with this axis, by mapping with axis's unique name.

INDEX.JS

```

var chartData = [
  { country: "USA", gold: 50, silver: 70, bronze: 45 },
  { country: "China", gold: 40, silver: 60, bronze: 55 },
  { country: "Japan", gold: 70, silver: 60, bronze: 50 },
  { country: "Australia", gold: 60, silver: 56, bronze: 40 },
  { country: "France", gold: 50, silver: 45, bronze: 35 },
  { country: "Germany", gold: 40, silver: 30, bronze: 22 },
  { country: "Italy", gold: 40, silver: 35, bronze: 37 },
  { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart3D = new ej.charts.Chart3D({
  primaryXAxis: {
    valueType: "Category"
  },
  axes:[
    {
      rowIndex: 0,
      name: 'yAxis',
    }
  ],
  series: [
    {
      dataSource: chartData,
      xName: 'country', yName: 'gold',
      type: 'Column'
    }, {
      dataSource: chartData,
      xName: 'country', yName: 'silver',
      yAxisName: 'yAxis',
      type: 'Column',
    }
  ],
  wallColor: 'transparent',
  enableRotation: true,
  rotation: 7,
  tilt: 10,
  depth: 100
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">

```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Inversed axis

<!-- markdownlint-disable MD033 -->

When an axis is inversed, highest value of the axis comes closer to origin and vice versa. To place an axis in inversed manner, set the [isInversed](#) property to **true**.

INDEX.JS

```

var chartData = [
    { country: "USA", gold: 50, silver: 70, bronze: 45 },
    { country: "China", gold: 40, silver: 60, bronze: 55 },
    { country: "Japan", gold: 70, silver: 60, bronze: 50 },
    { country: "Australia", gold: 60, silver: 56, bronze: 40 },
    { country: "France", gold: 50, silver: 45, bronze: 35 },
    { country: "Germany", gold: 40, silver: 30, bronze: 22 },
    { country: "Italy", gold: 40, silver: 35, bronze: 37 },
    { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart3D = new ej.charts.Chart3D({
    primaryXAxis: {
        valueType: "Category",
        isInversed: true
    },
    primaryYAxis: {
        isInversed: true
    },
    series: [
        {
            dataSource: chartData,
            xName: 'country', yName: 'gold',
            type: 'Column'
        }
    ]
});

```

```

    }],
    wallColor: 'transparent',
    enableRotation: true,
    rotation: 7,
    tilt: 10,
    depth: 100
  }, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Opposed position

To place an axis opposite from its original position, set the [opposedPosition](#) property to **true**.

INDEX.JS

```

var chartData = [
  { country: "USA", gold: 50, silver: 70, bronze: 45 },
  { country: "China", gold: 40, silver: 60, bronze: 55 },
  { country: "Japan", gold: 70, silver: 60, bronze: 50 },
  { country: "Australia", gold: 60, silver: 56, bronze: 40 },
  { country: "France", gold: 50, silver: 45, bronze: 35 },
  { country: "Germany", gold: 40, silver: 30, bronze: 22 },
  { country: "Italy", gold: 40, silver: 35, bronze: 37 },
  { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart3D = new ej.charts.Chart3D({
  primaryXAxis: {
    valueType: "Category"

```

```

    },
    primaryYAxis:
    {
        //Axis position as opposite
        opposedPosition: true
    },
    series: [
        {
            dataSource: chartData,
            xName: 'country', yName: 'gold',
            type: 'Column'
        }
    ],
    wallColor: 'transparent',
    enableRotation: true,
    rotation: 7,
    tilt: 10,
    depth: 100
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Multiple panes in EJ2 JavaScript 3D Chart control

The chart area can be divided into multiple panes using [rows](#) and [columns](#).

Rows

To split the chart area vertically into number of rows, use [rows](#) property of the 3D chart.

- The space for each row can be allocated by using the [height](#) property. The value can be either in percentage or in pixel.
- To associate a vertical axis to a particular row, specify its index to [rowIndex](#) property of the axis.

INDEX.JS

```

var chartData = [
  { x: 'Jan', y: 15, y1: 33 }, { x: 'Feb', y: 20, y1: 31 }, { x: 'Mar', y:
35, y1: 30 },
  { x: 'Apr', y: 40, y1: 28 }, { x: 'May', y: 80, y1: 29 }, { x: 'Jun', y:
70, y1: 30 },
  { x: 'Jul', y: 65, y1: 33 }, { x: 'Aug', y: 55, y1: 32 }, { x: 'Sep', y:
50, y1: 34 },
  { x: 'Oct', y: 30, y1: 32 }, { x: 'Nov', y: 35, y1: 32 }, { x: 'Dec', y:
35, y1: 31 }
];
var chart3D = new ej.charts.Chart3D({
  primaryXAxis: {
    title: 'Months',
    valueType: 'Category',
    interval: 1
  },
  primaryYAxis: {
    minimum: 0, maximum: 90, interval: 20,
    title: 'Temperature (Fahrenheit)',
    labelFormat: '{value}°F'
  },
  // Rows for chart axis
  rows:[
    {
      height: '50%'
    },{
      height: '50%'
    }
  ],
  axes:[
    {
      majorGridLines: { width: 0 },
      rowIndex: 1, opposedPosition: true,
      minimum: 24, maximum: 36, interval: 4,
      name: 'yAxis', title: 'Temperature (Celsius)',
      labelFormat: '{value}°C'
    }
  ],
  series:[{
    dataSource: chartData,
    xName: 'x', yName: 'y',
    name: 'Germany', type: 'Column'
  },{
    dataSource: chartData,
    xName: 'x', yName: 'y1', yAxisName: 'yAxis',
    name: 'Japan', type: 'Column'
  }],
  title: 'Weather Condition',
  wallColor: 'transparent',
  enableRotation: true,

```

```

    rotation: 7,
    tilt: 10,
    depth: 100
  }, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

For spanning the vertical axis along multiple rows, use [span](#) property of an axis.

INDEX.JS

```

var chartData = [
  { x: 'Jan', y: 15, y1: 33 }, { x: 'Feb', y: 20, y1: 31 }, { x: 'Mar', y:
35, y1: 30 },
  { x: 'Apr', y: 40, y1: 28 }, { x: 'May', y: 80, y1: 29 }, { x: 'Jun', y:
70, y1: 30 },
  { x: 'Jul', y: 65, y1: 33 }, { x: 'Aug', y: 55, y1: 32 }, { x: 'Sep', y:
50, y1: 34 },
  { x: 'Oct', y: 30, y1: 32 }, { x: 'Nov', y: 35, y1: 32 }, { x: 'Dec', y:
35, y1: 31 }
];
var chart3D = new ej.charts.Chart3D({
  primaryXAxis: {
    title: 'Months',
    valueType: 'Category',
    interval: 1
  },
  primaryYAxis: {

```

```

        minimum: 0, maximum: 90, interval: 20,
        title: 'Temperature (Fahrenheit)',
        labelFormat: '{value}°F',
        //Span for chart axis
        span: 2
    },
    // Rows for chart axis
    rows:[
        {
            height: '50%'
        },{
            height: '50%'
        }
    ],
    axes:[
        {
            majorGridLines: { width: 0 },
            rowIndex: 1, opposedPosition: true,
            minimum: 24, maximum: 36, interval: 4,
            name: 'yAxis', title: 'Temperature (Celsius)',
            labelFormat: '{value}°C'
        }
    ],
    series:[{
        dataSource: chartData,
        xName: 'x', yName: 'y',
        name: 'Germany', type: 'Column'
    },{
        dataSource: chartData,
        xName: 'x', yName: 'y1', yAxisName: 'yAxis',
        name: 'Japan', type: 'Column'
    }],
    title: 'Weather Condition',
    wallColor: 'transparent',
    enableRotation: true,
    rotation: 7,
    tilt: 10,
    depth: 100
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>

```

```
<body>

  <div id="container">
    <div id="element"></div>
  </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Columns

To split the chart area horizontally into number of columns, use [columns](#) property of the 3D chart.

- The space for each column can be allocated by using the [width](#) property. The given width can be either in percentage or in pixel.
- To associate a horizontal axis to a particular column, specify its index to [columnIndex](#) property of the axis.

INDEX.JS

```
var chartData = [
  { x: 'Jan', y: 15, y1: 33 }, { x: 'Feb', y: 20, y1: 31 }, { x: 'Mar', y:
35, y1: 30 },
  { x: 'Apr', y: 40, y1: 28 }, { x: 'May', y: 80, y1: 29 }, { x: 'Jun', y:
70, y1: 30 },
  { x: 'Jul', y: 65, y1: 33 }, { x: 'Aug', y: 55, y1: 32 }, { x: 'Sep', y:
50, y1: 34 },
  { x: 'Oct', y: 30, y1: 32 }, { x: 'Nov', y: 35, y1: 32 }, { x: 'Dec', y:
35, y1: 31 }
];
var chart3D = new ej.charts.Chart3D({
  primaryXAxis: {
    valueType: 'Category',
    interval: 1
  },
  primaryYAxis: {
    minimum: 0, maximum: 90, interval: 10,
    title: 'Temperature (Fahrenheit)',
    labelFormat: '{value}°F'
  },
  // Columns for chart axis
  columns: [
    {
      width: '50%'
    }, {
      width: '50%'
    }
  ],
  axes: [
    {
```



```

        majorGridLines: { width: 0 },
        columnIndex: 1,
        valueType: 'Category',
        name: 'xAxis'
    }
],
series:[{
    dataSource: chartData,
    xName: 'x', yName: 'y',
    name: 'Germany', type: 'Column'
},{
    dataSource: chartData,
    xName: 'x', yName: 'y1', xAxisName: 'xAxis',
    name: 'Japan', type: 'Column'
}],
title: 'Weather Condition',
wallColor: 'transparent',
enableRotation: true,
rotation: 7,
tilt: 10,
depth: 100
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

For spanning the vertical axis along multiple column, you can use [span](#) property of an axis.

INDEX.JS

```

var chartData = [
  { x: 'Jan', y: 15, y1: 33 }, { x: 'Feb', y: 20, y1: 31 }, { x: 'Mar', y:
35, y1: 30 },
  { x: 'Apr', y: 40, y1: 28 }, { x: 'May', y: 80, y1: 29 }, { x: 'Jun', y:
70, y1: 30 },
  { x: 'Jul', y: 65, y1: 33 }, { x: 'Aug', y: 55, y1: 32 }, { x: 'Sep', y:
50, y1: 34 },
  { x: 'Oct', y: 30, y1: 32 }, { x: 'Nov', y: 35, y1: 32 }, { x: 'Dec', y:
35, y1: 31 }
];
var chart3D = new ej.charts.Chart3D({
  primaryXAxis: {
    valueType: 'Category',
    interval: 1,
    // Span for chart axis
    span: 2
  },
  primaryYAxis: {
    minimum: 0, maximum: 90, interval: 10,
    title: 'Temperature (Fahrenheit)',
    labelFormat: '{value}°F'
  },
  // Columns for chart axis
  columns:[
    {
      width: '50%'
    },{
      width: '50%'
    }
  ],
  axes:[
    {
      majorGridLines: { width: 0 },
      columnIndex: 1,
      valueType: 'Category',
      name: 'xAxis'
    }
  ],
  series:[{
    dataSource: chartData,
    xName: 'x', yName: 'y',
    name: 'Germany', type: 'Column'
  },{
    dataSource: chartData,
    xName: 'x', yName: 'y1', xAxisName: 'xAxis',
    name: 'Japan', type: 'Column'
  }],
  title: 'Weather Condition',
  wallColor: 'transparent',
  enableRotation: true,
  rotation: 7,
  tilt: 10,
  depth: 100
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Chart Types

Column Chart in EJ2 JavaScript 3D Chart control

*Column chart***INDEX.JS**

```

var chartData = [
  { country: "USA", gold: 50, silver: 70, bronze: 45 },
  { country: "China", gold: 40, silver: 60, bronze: 55 },
  { country: "Japan", gold: 70, silver: 60, bronze: 50 },
  { country: "Australia", gold: 60, silver: 56, bronze: 40 },
  { country: "France", gold: 50, silver: 45, bronze: 35 },
  { country: "Germany", gold: 40, silver: 30, bronze: 22 },
  { country: "Italy", gold: 40, silver: 35, bronze: 37 },
  { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart3D = new ej.charts.Chart3D({
  primaryXAxis: {
    valueType: 'Category',
  },
  series:[{
    dataSource: chartData,
    xName: 'country', yName: 'gold',
    // Series type as column series
    type: 'Column'
  }],

```

```

    title: 'Olympic Medals',
    wallColor: 'transparent',
    enableRotation: true,
    rotation: 7,
    tilt: 10,
    depth: 100
  }, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Column space and width

The [columnSpacing](#) and [columnWidth](#) properties are used to customize the space between columns.

INDEX.JS

```

var chartData = [
  { country: "USA", gold: 50, silver: 70, bronze: 45 },
  { country: "China", gold: 40, silver: 60, bronze: 55 },
  { country: "Japan", gold: 70, silver: 60, bronze: 50 },
  { country: "Australia", gold: 60, silver: 56, bronze: 40 },
  { country: "France", gold: 50, silver: 45, bronze: 35 },
  { country: "Germany", gold: 40, silver: 30, bronze: 22 },
  { country: "Italy", gold: 40, silver: 35, bronze: 37 },
  { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart3D = new ej.charts.Chart3D({
  primaryXAxis: {
    valueType: 'Category',

```

```

    },
    series:[{
        dataSource: chartData,
        xName: 'country', yName: 'gold',
        type: 'Column'
    },
    {
        dataSource: chartData,
        xName: 'country',
        yName: 'silver',
        columnWidth: 0.75,
        columnSpacing: 0.5,
        type: 'Column',
    }],
    title: 'Olympic Medals',
    wallColor: 'transparent',
    enableRotation: true,
    rotation: 7,
    tilt: 10,
    depth: 100
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Grouped column

The data points can be grouped in the column type charts by using the [groupName](#) property. Data points with same group name are grouped together.

INDEX.JS

```

var chart3D = new ej.charts.Chart3D({
  primaryXAxis: {
    valueType: 'Category',
  },
  series: [
    {
      type: 'Column', xName: 'x', yName: 'y', groupName: 'USA',
      columnWidth: 0.7,
      dataSource: [{ x: '2012', y: 104 }, { x: '2016', y: 121 }, { x:
'2020', y: 113 }], columnSpacing: 0.1,
    },
    {
      type: 'Column', xName: 'x', yName: 'y', groupName: 'USA',
      columnWidth: 0.5,
      dataSource: [{ x: '2012', y: 46 }, { x: '2016', y: 46 }, { x:
'2020', y: 39 }], columnSpacing: 0.1,
    },
    {
      type: 'Column', xName: 'x', yName: 'y', groupName: 'UK',
      columnWidth: 0.7,
      dataSource: [{ x: '2012', y: 65 }, { x: '2016', y: 67 }, { x:
'2020', y: 65 }], columnSpacing: 0.1,
    },
    {
      type: 'Column', xName: 'x', yName: 'y', groupName: 'UK',
      columnWidth: 0.5,
      dataSource: [{ x: '2012', y: 29 }, { x: '2016', y: 27 }, { x:
'2020', y: 22 }], columnSpacing: 0.1,
    },
  ],
  wallColor: 'transparent',
  enableRotation: true,
  rotation: 7,
  tilt: 10,
  depth: 100
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```

```

    <div id="container">
      <div id="element"></div>
    </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Cylindrical column chart

To render a cylindrical column chart, set the [columnFacet](#) property to **Cylinder** in the chart series.

INDEX.JS

```

var chart3D = new ej.charts.Chart3D({
  primaryXAxis: {
    valueType: 'Category',
  },
  series: [
    {
      type: 'Column', columnFacet: 'Cylinder', xName: 'x', yName: 'y',
      columnWidth: 0.9,
      dataSource: [{ x: 'Czechia', y: 1.11 }, { x: 'Spain', y: 1.66 },
        { x: 'USA', y: 1.56 }, { x: 'Germany', y: 3.1 }, { x: 'Russia', y: 1.35 },
        { x: 'Slovakia', y: 1 }, { x: 'South Korea', y: 3.16 }, { x: 'France', y:
        0.92 }],
    },
  ],
  wallColor: 'transparent',
  enableRotation: true,
  rotation: 7,
  tilt: 10,
  depth: 100
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>

```

```
<body>

  <div id="container">
    <div id="element"></div>
  </div>
</script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Series customization

The following properties can be used to customize the **column** series.

- [fill](#) – Specifies the color of the series.
- [opacity](#) – Specifies the opacity of the [fill](#) color.

INDEX.JS

```
var chartData = [
  { country: "USA", gold: 50, silver: 70, bronze: 45 },
  { country: "China", gold: 40, silver: 60, bronze: 55 },
  { country: "Japan", gold: 70, silver: 60, bronze: 50 },
  { country: "Australia", gold: 60, silver: 56, bronze: 40 },
  { country: "France", gold: 50, silver: 45, bronze: 35 },
  { country: "Germany", gold: 40, silver: 30, bronze: 22 },
  { country: "Italy", gold: 40, silver: 35, bronze: 37 },
  { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart3D = new ej.charts.Chart3D({
  primaryXAxis: {
    valueType: 'Category',
  },
  series: [
    {
      dataSource: chartData,
      xName: 'country', yName: 'gold',
      type: 'Column', fill: 'red'
    },
  ],
  wallColor: 'transparent',
  enableRotation: true,
  rotation: 7,
  tilt: 10,
  depth: 100
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
```



```

<title>EJ2 Animation</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Stacked column chart in EJ2 JavaScript 3D Chart control

Stacked column chart

To render a stacked column series, use series [type](#) as `StackingColumn` and inject `StackingColumnSeries3D` module using `Chart3D.Inject(StackingColumnSeries3D)` method.

INDEX.JS

```

var stackedData = [
    { x: 2000, y: 0.61, y1: 0.03, y2: 0.48 }, { x: 2001, y: 0.81, y1: 0.05,
y2: 0.53 },
    { x: 2002, y: 0.91, y1: 0.06, y2: 0.57 }, { x: 2003, y: 1, y1: 0.09, y2:
0.61 }, { x: 2004, y: 1.19, y1: 0.14, y2: 0.63 },
    { x: 2005, y: 1.47, y1: 0.20, y2: 0.64 }, { x: 2006, y: 1.74, y1: 0.29,
y2: 0.66 }, { x: 2007, y: 1.98, y1: 0.46, y2: 0.76 },
    { x: 2008, y: 1.99, y1: 0.64, y2: 0.77 }, { x: 2009, y: 1.70, y1: 0.75,
y2: 0.55 }
];
var chart3D = new ej.charts.Chart3D({
    primaryXAxis: {
        valueType: 'Category',
    },
    series: [
        {
            dataSource: stackedData, xName: 'x', yName: 'y',
            //Series type as stacked column
            type: 'StackingColumn'
        }, {
            dataSource: stackedData, xName: 'x', yName: 'y1',

```

```

        type: 'StackingColumn'
    }, {
        dataSource: stackedData, xName: 'x', yName: 'y2',
        type: 'StackingColumn'
    }
],
wallColor: 'transparent',
enableRotation: true,
rotation: 7,
tilt: 10,
depth: 100
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Stacking group

To group the stacked column, the [stackingGroup](#) property can be used. The columns with same group name are stacked on top of each other.

INDEX.JS

```

var stackedData = [
  { x: 2000, y: 0.61, y1: 0.03, y2: 0.48 }, { x: 2001, y: 0.81, y1: 0.05,
  y2: 0.53 },
  { x: 2002, y: 0.91, y1: 0.06, y2: 0.57 }, { x: 2003, y: 1, y1: 0.09, y2:
  0.61 }, { x: 2004, y: 1.19, y1: 0.14, y2: 0.63 },

```

```

    { x: 2005, y: 1.47, y1: 0.20, y2: 0.64 }, { x: 2006, y: 1.74, y1: 0.29,
y2: 0.66 }, { x: 2007, y: 1.98, y1: 0.46, y2: 0.76 },
    { x: 2008, y: 1.99, y1: 0.64, y2: 0.77 }, { x: 2009, y: 1.70, y1: 0.75,
y2: 0.55 }
];
var chart3D = new ej.charts.Chart3D({
    primaryXAxis: {
        valueType: 'Category',
    },
    series: [
        {
            dataSource: stackedData, xName: 'x', yName: 'y',
            //Series type as stacked column
            type: 'StackingColumn', stackingGroup: 'UKAndGermany'
        }, {
            dataSource: stackedData, xName: 'x', yName: 'y1',
            type: 'StackingColumn', stackingGroup: 'UKAndGermany'
        }, {
            dataSource: stackedData, xName: 'x', yName: 'y2',
            type: 'StackingColumn', stackingGroup: 'UKAndGermany'
        }
    ],
    wallColor: 'transparent',
    enableRotation: true,
    rotation: 7,
    tilt: 10,
    depth: 100
}, '#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
```

```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Cylindrical stacked column chart

To render a cylindrical stacked column chart, set the `columnFacet` property to `Cylinder` in the chart series.

INDEX.JS

```
var stackedData = [
  { x: 2000, y: 0.61, y1: 0.03, y2: 0.48 }, { x: 2001, y: 0.81, y1: 0.05,
y2: 0.53 },
  { x: 2002, y: 0.91, y1: 0.06, y2: 0.57 }, { x: 2003, y: 1, y1: 0.09, y2:
0.61 }, { x: 2004, y: 1.19, y1: 0.14, y2: 0.63 },
  { x: 2005, y: 1.47, y1: 0.20, y2: 0.64 }, { x: 2006, y: 1.74, y1: 0.29,
y2: 0.66 }, { x: 2007, y: 1.98, y1: 0.46, y2: 0.76 },
  { x: 2008, y: 1.99, y1: 0.64, y2: 0.77 }, { x: 2009, y: 1.70, y1: 0.75,
y2: 0.55 }
];
var chart3D = new ej.charts.Chart3D({
  primaryXAxis: {
    valueType: 'Category',
  },
  series: [
    {
      dataSource: stackedData, xName: 'x', yName: 'y',
      //Series type as stacked column
      type: 'StackingColumn', columnFacet: 'Cylinder'
    }, {
      dataSource: stackedData, xName: 'x', yName: 'y1',
      type: 'StackingColumn', columnFacet: 'Cylinder'
    }, {
      dataSource: stackedData, xName: 'x', yName: 'y2',
      type: 'StackingColumn', columnFacet: 'Cylinder'
    }
  ],
  wallColor: 'transparent',
  enableRotation: true,
  rotation: 7,
  tilt: 10,
  depth: 100
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Series customization

The following properties can be used to customize the **stacked column** series.

- [fill](#) – Specifies the color of the series.
- [opacity](#) – Specifies the opacity of the [fill](#) color.

INDEX.JS

```

var stackedData = [
    { x: 2000, y: 0.61, y1: 0.03, y2: 0.48 }, { x: 2001, y: 0.81, y1: 0.05,
y2: 0.53 },
    { x: 2002, y: 0.91, y1: 0.06, y2: 0.57 }, { x: 2003, y: 1, y1: 0.09, y2:
0.61 }, { x: 2004, y: 1.19, y1: 0.14, y2: 0.63 },
    { x: 2005, y: 1.47, y1: 0.20, y2: 0.64 }, { x: 2006, y: 1.74, y1: 0.29,
y2: 0.66 }, { x: 2007, y: 1.98, y1: 0.46, y2: 0.76 },
    { x: 2008, y: 1.99, y1: 0.64, y2: 0.77 }, { x: 2009, y: 1.70, y1: 0.75,
y2: 0.55 }
];
var chart3D = new ej.charts.Chart3D({
    primaryXAxis: {
        valueType: 'Category',
    },
    series: [
        {
            dataSource: stackedData, xName: 'x', yName: 'y',
            //Series type as stacked column
            type: 'StackingColumn', fill: "green"
        }, {
            dataSource: stackedData, xName: 'x', yName: 'y1',
            type: 'StackingColumn', fill: "red"
        }, {
            dataSource: stackedData, xName: 'x', yName: 'y2',
            type: 'StackingColumn', fill: "yellow"
        }
    ],
    wallColor: 'transparent',

```

```

    enableRotation: true,
    rotation: 7,
    tilt: 10,
    depth: 100
  }, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

100% Stacked column chart in EJ2 JavaScript 3D Chart control

100% Stacked column chart

INDEX.JS

```

var chartData = [
  { x: '2013', y: 9628912, y1: 4298390, y2: 2842133, y3: 2006366 },
  { x: '2014', y: 9609326, y1: 4513769, y2: 3016710, y3: 2165566 },
  { x: '2015', y: 7485587, y1: 4543838, y2: 3034081, y3: 2279503 },
  { x: '2016', y: 7793066, y1: 4999266, y2: 2945295, y3: 2359756 },
  { x: '2017', y: 6856880, y1: 5235842, y2: 3302336, y3: 2505741 }
];
var chart3D = new ej.charts.Chart3D({
  primaryXAxis: {
    valueType: 'Category',
  },
  //Initializing Primary Y Axis
  primaryYAxis: {
    {
      interval: 25
    }
  }
});

```

```

    },
    //Initializing Chart Series
    series: [
        {
            dataSource: chartData, xName: 'x', yName: 'y',
            type: 'StackingColumn100',
            name: 'General Motors'
        }, {
            dataSource: chartData, xName: 'x', yName: 'y1',
            type: 'StackingColumn100', name: 'Honda'
        }, {
            dataSource: chartData, xName: 'x', yName: 'y2',
            type: 'StackingColumn100', name: 'Suzuki'
        }
    ],
    enableRotation: true,
    rotation: 7,
    tilt: 10,
    depth: 100,
    wallColor: 'transparent',
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

100% Cylindrical stacked column chart

To render a 100% cylindrical stacked column chart, set the [columnFacet](#) property to `Cylinder` in the chart series.

INDEX.JS

```

var chartData = [
  { x: '2013', y: 9628912, y1: 4298390, y2: 2842133, y3: 2006366 },
  { x: '2014', y: 9609326, y1: 4513769, y2: 3016710, y3: 2165566 },
  { x: '2015', y: 7485587, y1: 4543838, y2: 3034081, y3: 2279503 },
  { x: '2016', y: 7793066, y1: 4999266, y2: 2945295, y3: 2359756 },
  { x: '2017', y: 6856880, y1: 5235842, y2: 3302336, y3: 2505741 }
];
var chart3D = new ej.charts.Chart3D({
  primaryXAxis: {
    valueType: 'Category',
  },
  //Initializing Primary Y Axis
  primaryYAxis: {
    {
      interval: 25
    },
  },
  //Initializing Chart Series
  series: [
    {
      dataSource: chartData, xName: 'x', yName: 'y',
      type: 'StackingColumn100',
      columnFacet: 'Cylinder',
      name: 'General Motors'
    }, {
      dataSource: chartData, xName: 'x', yName: 'y1',
      type: 'StackingColumn100', columnFacet: 'Cylinder', name:
'Honda'
    }, {
      dataSource: chartData, xName: 'x', yName: 'y2',
      type: 'StackingColumn100', columnFacet: 'Cylinder', name:
'Suzuki'
    }
  ],
  enableRotation: true,
  rotation: 7,
  tilt: 10,
  depth: 100,
  wallColor: 'transparent',
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>

```



```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Series customization

The following properties can be used to customize the **100% stacked column** series.

- [fill](#) – Specifies the color of the series.
- [opacity](#) – Specifies the opacity of the [fill](#) color.

INDEX.JS

```

var chartData = [
    { x: '2013', y: 9628912, y1: 4298390, y2: 2842133, y3: 2006366 },
    { x: '2014', y: 9609326, y1: 4513769, y2: 3016710, y3: 2165566 },
    { x: '2015', y: 7485587, y1: 4543838, y2: 3034081, y3: 2279503 },
    { x: '2016', y: 7793066, y1: 4999266, y2: 2945295, y3: 2359756 },
    { x: '2017', y: 6856880, y1: 5235842, y2: 3302336, y3: 2505741 }
];
var chart3D = new ej.charts.Chart3D({
    primaryXAxis: {
        valueType: 'Category',
    },
    primaryYAxis: {
        interval: 25
    },
    series: [
        {
            dataSource: chartData, xName: 'x', yName: 'y',
            type: 'StackingColumn100',
            fill: 'red',
            name: 'General Motors'
        }, {
            dataSource: chartData, xName: 'x', yName: 'y1',
            type: 'StackingColumn100', fill: 'green', name: 'Honda'
        }, {
            dataSource: chartData, xName: 'x', yName: 'y2',
            type: 'StackingColumn100', fill: 'yellow', name: 'Suzuki'
        }
    ],
    enableRotation: true,

```

```

rotation: 7,
tilt: 10,
depth: 100,
wallColor: 'transparent',
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Bar Chart in EJ2 JavaScript 3D Chart control

Bar chart

INDEX.JS

```

var chartData = [
  { country: "USA", gold: 50, silver: 70, bronze: 45 },
  { country: "China", gold: 40, silver: 60, bronze: 55 },
  { country: "Japan", gold: 70, silver: 60, bronze: 50 },
  { country: "Australia", gold: 60, silver: 56, bronze: 40 },
  { country: "France", gold: 50, silver: 45, bronze: 35 },
  { country: "Germany", gold: 40, silver: 30, bronze: 22 },
  { country: "Italy", gold: 40, silver: 35, bronze: 37 },
  { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart3D = new ej.charts.Chart3D({
  primaryXAxis: {
    valueType: 'Category',
  },
  series: [

```

```

        {
            dataSource: chartData,
            xName: 'country', yName: 'gold',
            type: 'Bar'
        },
    ],
    wallColor: 'transparent',
    enableRotation: true,
    rotation: 7,
    tilt: 10,
    depth: 100
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Bar space and width

The [columnSpacing](#) and [columnWidth](#) properties are used to customize the space between bars.

INDEX.JS

```

var chartData = [
  { country: "USA", gold: 50, silver: 70, bronze: 45 },
  { country: "China", gold: 40, silver: 60, bronze: 55 },
  { country: "Japan", gold: 70, silver: 60, bronze: 50 },
  { country: "Australia", gold: 60, silver: 56, bronze: 40 },
  { country: "France", gold: 50, silver: 45, bronze: 35 },
  { country: "Germany", gold: 40, silver: 30, bronze: 22 },
  { country: "Italy", gold: 40, silver: 35, bronze: 37 },

```

```

    { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
  ];
  var chart3D = new ej.charts.Chart3D({
    primaryXAxis: {
      valueType: 'Category',
    },
    series: [
      {
        dataSource: chartData,
        xName: 'country', yName: 'gold',
        type: 'Bar'
      },
      {
        dataSource: chartData,
        xName: 'country',
        yName: 'silver',
        columnSpacing: 0.5,
        columnWidth: 0.75,
        // Series type as bar series
        type: 'Bar',
      }
    ],
    wallColor: 'transparent',
    enableRotation: true,
    rotation: 7,
    tilt: 10,
    depth: 100
  }, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
  var ele = document.getElementById('container');
  if(ele) {
    ele.style.visibility = "visible";
  }
  </script>

```

```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Grouped bar

The data points can be grouped in the bar type charts by using the [groupName](#) property. Data points with same group name are grouped together.

INDEX.JS

```
var chart3D = new ej.charts.Chart3D({
  primaryXAxis: {
    valueType: 'Category',
  },
  series: [
    {
      type: 'Bar', xName: 'x', yName: 'y', groupName: 'USA',
      columnWidth: 0.7,
      dataSource: [{ x: '2012', y: 104 }, { x: '2016', y: 121 }, { x:
        '2020', y: 113 }], columnSpacing: 0.1,
    },
    {
      type: 'Bar', xName: 'x', yName: 'y', groupName: 'USA',
      columnWidth: 0.5,
      dataSource: [{ x: '2012', y: 46 }, { x: '2016', y: 46 }, { x:
        '2020', y: 39 }], columnSpacing: 0.1,
    },
    {
      type: 'Bar', xName: 'x', yName: 'y', groupName: 'UK',
      columnWidth: 0.7,
      dataSource: [{ x: '2012', y: 65 }, { x: '2016', y: 67 }, { x:
        '2020', y: 65 }], columnSpacing: 0.1,
    },
    {
      type: 'Bar', xName: 'x', yName: 'y', groupName: 'UK',
      columnWidth: 0.5,
      dataSource: [{ x: '2012', y: 29 }, { x: '2016', y: 27 }, { x:
        '2020', y: 22 }], columnSpacing: 0.1,
    },
  ],
  wallColor: 'transparent',
  enableRotation: true,
  rotation: 7,
  tilt: 10,
  depth: 100
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Cylindrical bar chart

To render a cylindrical bar chart, set the [columnFacet](#) property to **Cylinder** in the chart series.

INDEX.JS

```

var chartData = [
    { country: "USA", gold: 50, silver: 70, bronze: 45 },
    { country: "China", gold: 40, silver: 60, bronze: 55 },
    { country: "Japan", gold: 70, silver: 60, bronze: 50 },
    { country: "Australia", gold: 60, silver: 56, bronze: 40 },
    { country: "France", gold: 50, silver: 45, bronze: 35 },
    { country: "Germany", gold: 40, silver: 30, bronze: 22 },
    { country: "Italy", gold: 40, silver: 35, bronze: 37 },
    { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart3D = new ej.charts.Chart3D({
    primaryXAxis: {
        valueType: 'Category',
    },
    series: [
        {
            dataSource: chartData,
            xName: 'country', yName: 'gold',
            type: 'Bar', columnFacet: 'Cylinder'
        },
    ],
    wallColor: 'transparent',
    enableRotation: true,
    rotation: 7,
    tilt: 10,
    depth: 100
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Series customization

The following properties can be used to customize the **bar** series.

- [fill](#) – Specifies the color of the series.
- [opacity](#) – Specifies the opacity of the [fill](#) color.

INDEX.JS

```

var numData = [
  { x: 2005, y: 28 }, { x: 2006, y: 25 },
  { x: 2007, y: 26 }, { x: 2008, y: 27 },
  { x: 2009, y: 32 }, { x: 2010, y: 35 }, { x: 2011, y: 30 }];
var chart3D = new ej.charts.Chart3D({
  primaryXAxis: {
    valueType: 'Category',
  },
  series: [
    {
      dataSource: numData,
      xName: 'x',
      yName: 'y',
      opacity: 0.5,
      fill: 'blue',
      // Series type as bar series
      type: 'Bar',
    },
  ],
});

```

```

    wallColor: 'transparent',
    enableRotation: true,
    rotation: 7,
    tilt: 10,
    depth: 100
  }, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Stacked bar chart in EJ2 JavaScript 3D Chart control

Stacked bar chart

To render a stacked bar series, use series [type](#) as `StackingBar` and inject `StackingBarSeries3D` module using `Chart3D.Inject(StackingBarSeries3D)` method.

INDEX.JS

```

var chart3D = new ej.charts.Chart3D({
  primaryXAxis: {
    valueType: 'Category'
  },
  //Initializing Primary Y Axis
  primaryYAxis: {
    {
      interval: 20
    }
  },
  //Initializing Chart Series
  series: [

```



```

    {
      type: 'StackingBar',
      dataSource: [
        { x: 'Sochi', y: 9 },
        { x: 'Rio', y: 46 },
        { x: 'Pyeongchang', y: 9 },
        { x: 'Tokyo', y: 39 },
        { x: 'Beijing', y: 8 },
      ],
      name: 'America',
      xName: 'x',
      yName: 'y'
    },
    {
      type: 'StackingBar',
      dataSource: [
        { x: 'Sochi', y: 10 },
        { x: 'Rio', y: 4 },
        { x: 'Pyeongchang', y: 11 },
        { x: 'Tokyo', y: 7 },
        { x: 'Beijing', y: 4 },],
      name: 'Canada',
      xName: 'x',
      yName: 'y'
    },
    {
      type: 'StackingBar',
      dataSource: [
        { x: 'Sochi', y: 4 },
        { x: 'Rio', y: 10 },
        { x: 'Pyeongchang', y: 5 },
        { x: 'Tokyo', y: 10 },
        { x: 'Beijing', y: 5 },],
      name: 'France',
      xName: 'x',
      yName: 'y'
    }
  ],
  enableRotation: true,
  rotation: 22,
  depth: 100,
  legendSettings: {
    enableHighlight: true
  },
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Stacking group

To group the stacked bar, the [stackingGroup](#) property can be used. The columns with same group name are stacked on top of each other.

INDEX.JS

```

var chart3D = new ej.charts.Chart3D({
    primaryXAxis: {
        valueType: 'Category'
    },
    //Initializing Primary Y Axis
    primaryYAxis: {
        interval: 20
    },
    //Initializing Chart Series
    series: [
        {
            type: 'StackingBar',
            dataSource: [
                { x: 'Sochi', y: 9 },
                { x: 'Rio', y: 46 },
                { x: 'Pyeongchang', y: 9 },
                { x: 'Tokyo', y: 39 },
                { x: 'Beijing', y: 8 },
            ],
            stackingGroup: 'JohnAndAndrew',
            name: 'America',
            xName: 'x',
            yName: 'y'
        },
        {
            type: 'StackingBar',
            dataSource: [
                { x: 'Sochi', y: 10 },
                { x: 'Rio', y: 4 },
            ]
        }
    ]
});

```

```

        { x: 'Pyeongchang', y: 11 },
        { x: 'Tokyo', y: 7 },
        { x: 'Beijing', y: 4 },],
        name: 'Canada',
        stackingGroup: 'JohnAndAndrew',
        xName: 'x',
        yName: 'y'
    },
    {
        type: 'StackingBar',
        dataSource: [
            { x: 'Sochi', y: 4 },
            { x: 'Rio', y: 10 },
            { x: 'Pyeongchang', y: 5 },
            { x: 'Tokyo', y: 10 },
            { x: 'Beijing', y: 5 },],
        name: 'France',
        stackingGroup: 'ThomasAndMichael',
        xName: 'x',
        yName: 'y'
    }
],
enableRotation: true,
rotation: 22,
depth: 100,
legendSettings: {
    enableHighlight: true
},
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}

```

```

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Cylindrical stacked bar chart

To render a cylindrical stacked bar chart, set the [columnFacet](#) property to **Cylinder** in the chart series.

INDEX.JS

```

var chart3D = new ej.charts.Chart3D({
  primaryXAxis: {
    valueType: 'Category'
  },
  //Initializing Primary Y Axis
  primaryYAxis: {
    interval: 20
  },
  //Initializing Chart Series
  series: [
    {
      type: 'StackingBar',
      dataSource: [
        { x: 'Sochi', y: 9 },
        { x: 'Rio', y: 46 },
        { x: 'Pyeongchang', y: 9 },
        { x: 'Tokyo', y: 39 },
        { x: 'Beijing', y: 8 },
      ],
      columnFacet: "Cylinder",
      name: 'America',
      xName: 'x',
      yName: 'y'
    },
    {
      type: 'StackingBar',
      dataSource: [
        { x: 'Sochi', y: 10 },
        { x: 'Rio', y: 4 },
        { x: 'Pyeongchang', y: 11 },
        { x: 'Tokyo', y: 7 },
        { x: 'Beijing', y: 4 },
      ],
      name: 'Canada',
      columnFacet: "Cylinder",
      xName: 'x',
      yName: 'y'
    },
    {
      type: 'StackingBar',
      dataSource: [
        { x: 'Sochi', y: 4 },
        { x: 'Rio', y: 10 },
        { x: 'Pyeongchang', y: 5 },
        { x: 'Tokyo', y: 10 },
        { x: 'Beijing', y: 5 },
      ],
      name: 'France',
    }
  ]
});

```

```

        columnFacet: "Cylinder",
        xName: 'x',
        yName: 'y'
    }
],
enableRotation: true,
rotation: 22,
depth: 100,
legendSettings: {
    enableHighlight: true
},
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Series customization

The following properties can be used to customize the **stacked bar** series.

- [fill](#) – Specifies the color of the series.
- [opacity](#) – Specifies the opacity of the [fill](#) color.

INDEX.JS

```

var chart3D = new ej.charts.Chart3D({
    primaryXAxis: {
        valueType: 'Category'
    }
});

```

```

    },
    primaryYAxis:
    {
        interval: 20
    },
    series: [
        {
            type: 'StackingBar',
            dataSource: [
                { x: 'Sochi', y: 9 },
                { x: 'Rio', y: 46 },
                { x: 'Pyeongchang', y: 9 },
                { x: 'Tokyo', y: 39 },
                { x: 'Beijing', y: 8 },
            ],
            fill: "red",
            name: 'America',
            xName: 'x',
            yName: 'y'
        },
        {
            type: 'StackingBar',
            dataSource: [
                { x: 'Sochi', y: 10 },
                { x: 'Rio', y: 4 },
                { x: 'Pyeongchang', y: 11 },
                { x: 'Tokyo', y: 7 },
                { x: 'Beijing', y: 4 },],
            name: 'Canada',
            fill: "green",
            xName: 'x',
            yName: 'y'
        },
        {
            type: 'StackingBar',
            dataSource: [
                { x: 'Sochi', y: 4 },
                { x: 'Rio', y: 10 },
                { x: 'Pyeongchang', y: 5 },
                { x: 'Tokyo', y: 10 },
                { x: 'Beijing', y: 5 },],
            name: 'France',
            fill: "yellow",
            xName: 'x',
            yName: 'y'
        }
    ],
    enableRotation: true,
    rotation: 22,
    depth: 100,
    legendSettings: {
        enableHighlight: true
    },
}, '#element');

```

[INDEX.HTML](#)

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

100% Stacked bar chart in EJ2 JavaScript 3D Chart control

100% Stacked bar chart

INDEX.JS

```

var chartData = [
  { x: '2013', y: 9628912, y1: 4298390, y2: 2842133, y3: 2006366 },
  { x: '2014', y: 9609326, y1: 4513769, y2: 3016710, y3: 2165566 },
  { x: '2015', y: 7485587, y1: 4543838, y2: 3034081, y3: 2279503 },
  { x: '2016', y: 7793066, y1: 4999266, y2: 2945295, y3: 2359756 },
  { x: '2017', y: 6856880, y1: 5235842, y2: 3302336, y3: 2505741 }
];
var chart3D = new ej.charts.Chart3D({
  primaryXAxis: {
    valueType: 'Category',
  },
  //Initializing Primary Y Axis
  primaryYAxis: {
    {
      interval: 25
    },
  },
  //Initializing Chart Series
  series: [
    {
      dataSource: chartData, xName: 'x', yName: 'y',
      type: 'StackingColumn100',
      name: 'General Motors'
    }, {

```

```

        dataSource: chartData, xName: 'x', yName: 'y1',
        type: 'StackingColumn100', name: 'Honda'
    }, {
        dataSource: chartData, xName: 'x', yName: 'y2',
        type: 'StackingColumn100', name: 'Suzuki'
    }
],
enableRotation: true,
rotation: 7,
tilt: 10,
depth: 100,
wallColor: 'transparent',
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

100% Cylindrical stacked bar chart

To render a cylindrical 100% stacked bar chart, set the [columnFacet](#) property to **Cylinder** in the chart series.

INDEX.JS

```

var chartData = [
  { x: '2013', y: 9628912, y1: 4298390, y2: 2842133, y3: 2006366 },
  { x: '2014', y: 9609326, y1: 4513769, y2: 3016710, y3: 2165566 },
  { x: '2015', y: 7485587, y1: 4543838, y2: 3034081, y3: 2279503 },
  { x: '2016', y: 7793066, y1: 4999266, y2: 2945295, y3: 2359756 },

```



```

    { x: '2017', y: 6856880, y1: 5235842, y2: 3302336, y3: 2505741 }
  ];
  var chart3D = new ej.charts.Chart3D({
    primaryXAxis: {
      valueType: 'Category',
    },
    primaryYAxis: {
      interval: 20
    },
    series: [
      {
        dataSource: chartData, xName: 'x', yName: 'y',
        type: 'StackingBar100', columnFacet: "Cylinder",
        name: 'General Motors'
      }, {
        dataSource: chartData, xName: 'x', yName: 'y1',
        type: 'StackingBar100', columnFacet: "Cylinder", name: 'Honda'
      }, {
        dataSource: chartData, xName: 'x', yName: 'y2',
        type: 'StackingBar100', columnFacet: "Cylinder", name: 'Suzuki'
      }, {
        dataSource: chartData, xName: 'x', yName: 'y3',
        type: 'StackingBar100', name: 'BMW', columnFacet: "Cylinder",
      }
    ],
    enableRotation: true,
    rotation: 22,
    depth: 100,
    wallColor: 'transparent',
  }, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {

```

```

    ele.style.visibility = "visible";
  }
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Series customization

The following properties can be used to customize the **100% stacked bar** series.

- [fill](#) – Specifies the color of the series.
- [opacity](#) – Specifies the opacity of the [fill](#) color.

INDEX.JS

```

var chartData = [
  { x: '2013', y: 9628912, y1: 4298390, y2: 2842133, y3: 2006366 },
  { x: '2014', y: 9609326, y1: 4513769, y2: 3016710, y3: 2165566 },
  { x: '2015', y: 7485587, y1: 4543838, y2: 3034081, y3: 2279503 },
  { x: '2016', y: 7793066, y1: 4999266, y2: 2945295, y3: 2359756 },
  { x: '2017', y: 6856880, y1: 5235842, y2: 3302336, y3: 2505741 }
];
var chart3D = new ej.charts.Chart3D({
  primaryXAxis: {
    valueType: 'Category',
  },
  primaryYAxis: {
    interval: 20
  },
  series: [
    {
      dataSource: chartData, xName: 'x', yName: 'y',
      type: 'StackingBar100', fill: "red",
      name: 'General Motors'
    }, {
      dataSource: chartData, xName: 'x', yName: 'y1',
      type: 'StackingBar100', fill: "green", name: 'Honda'
    }, {
      dataSource: chartData, xName: 'x', yName: 'y2',
      type: 'StackingBar100', fill: "yellow", name: 'Suzuki'
    }, {
      dataSource: chartData, xName: 'x', yName: 'y3',
      type: 'StackingBar100', name: 'BMW', fill: "blue"
    }
  ],
  enableRotation: true,
  rotation: 22,
  depth: 100,
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>

```

```

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Data labels in EJ2 JavaScript 3D Chart control

Data labels are fields that includes information about the sample point connected to an output. It can be added to a chart series by enabling the [visible](#) property in the [dataLabel](#). By default, the labels will arrange smartly without overlapping.

INDEX.JS

```

var chartData = [
    { country: "USA", gold: 50, silver: 70, bronze: 45 },
    { country: "China", gold: 40, silver: 60, bronze: 55 },
    { country: "Japan", gold: 70, silver: 60, bronze: 50 },
    { country: "Australia", gold: 60, silver: 56, bronze: 40 },
    { country: "France", gold: 50, silver: 45, bronze: 35 },
    { country: "Germany", gold: 40, silver: 30, bronze: 22 },
    { country: "Italy", gold: 40, silver: 35, bronze: 37 },
    { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart3D = new ej.charts.Chart3D({
    primaryXAxis: {
        valueType: 'Category',
    },
    series:[{
        dataSource: chartData,
        xName: 'country', yName: 'gold',
        // Series type as column series
        type: 'Column',
        dataLabel: {
            visible: true
        }
    }
]
},

```

```

    title: 'Olympic Medals',
    wallColor: 'transparent',
    enableRotation: true,
    rotation: 7,
    tilt: 10,
    depth: 100
  }, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Note: To use data label feature, we need to inject `DataLabel3D` module using `Chart3D.Inject(DataLabel3D)` method.

Position

The [position](#) property is used to place the label either on `Top`, `Middle`, or `Bottom`.

INDEX.JS

```

var chartData = [
  { country: "USA", gold: 50, silver: 70, bronze: 45 },
  { country: "China", gold: 40, silver: 60, bronze: 55 },
  { country: "Japan", gold: 70, silver: 60, bronze: 50 },
  { country: "Australia", gold: 60, silver: 56, bronze: 40 },
  { country: "France", gold: 50, silver: 45, bronze: 35 },
  { country: "Germany", gold: 40, silver: 30, bronze: 22 },
  { country: "Italy", gold: 40, silver: 35, bronze: 37 },
  { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
]

```

```

];
var chart3D = new ej.charts.Chart3D({
  primaryXAxis: {
    valueType: 'Category',
  },
  series:[{
    dataSource: chartData,
    xName: 'country', yName: 'gold',
    // Series type as column series
    type: 'Column',
    dataLabel: {
      visible: true,
      position: 'Middle'
    }
  }],
  title: 'Olympic Medals',
  wallColor: 'transparent',
  enableRotation: true,
  rotation: 7,
  tilt: 10,
  depth: 100
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Template

Label content can be formatted by using the template option. Inside the template, the placeholder text `${point.x}` and `${point.y}` can be added to display corresponding data points x & y value. Using [template](#) property, the data label template can be set.

INDEX.JS

```
var chartData = [
  { country: "USA", gold: 50, silver: 70, bronze: 45 },
  { country: "China", gold: 40, silver: 60, bronze: 55 },
  { country: "Japan", gold: 70, silver: 60, bronze: 50 },
  { country: "Australia", gold: 60, silver: 56, bronze: 40 },
  { country: "France", gold: 50, silver: 45, bronze: 35 },
  { country: "Germany", gold: 40, silver: 30, bronze: 22 },
  { country: "Italy", gold: 40, silver: 35, bronze: 37 },
  { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart3D = new ej.charts.Chart3D({
  primaryXAxis: {
    valueType: 'Category',
  },
  series:[{
    dataSource: chartData,
    xName: 'country', yName: 'gold',
    // Series type as column series
    type: 'Column',
    dataLabel: {
      visible: true,
      template: '<div style="border: 1px solid black; padding: 3px 3px 3px 3px"><div>${point.x}</div><div>${point.y}</div></div>'
    }
  }],
  title: 'Olympic Medals',
  wallColor: 'transparent',
  enableRotation: true,
  rotation: 7,
  tilt: 10,
  depth: 100
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
```

```
<body>

  <div id="container">
    <div id="element"></div>
  </div>
</body>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Text mapping

Text from the data source can be mapped using the [name](#) property.

INDEX.JS

```
var chart3D = new ej.charts.Chart3D({
  primaryXAxis: {
    valueType: 'Category',
  },
  series:[{
    dataSource: [{ x: 'Jan', y: 12, text: 'January : 12°C' }, { x:
'Feb', y: 8, text: 'February : 8°C' }, { x: 'Mar', y: 11, text: 'March :
11°C' }, { x: 'Apr', y: 6, text: 'April : 6°C' }],
    xName: 'x', yName: 'y',
    // Series type as column series
    type: 'Column',
    dataLabel: {
      visible: true,
      name: "text"
    }
  }],
  title: 'Olympic Medals',
  wallColor: 'transparent',
  enableRotation: true,
  rotation: 7,
  tilt: 10,
  depth: 100
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Format

Data label for the chart can be formatted using the [format](#) property. The global formatting options can be used as 'n', 'p', and 'c'.

INDEX.JS

```

var chartData = [
    { country: "USA", gold: 50, silver: 70, bronze: 45 },
    { country: "China", gold: 40, silver: 60, bronze: 55 },
    { country: "Japan", gold: 70, silver: 60, bronze: 50 },
    { country: "Australia", gold: 60, silver: 56, bronze: 40 }
];
var chart3D = new ej.charts.Chart3D({
    primaryXAxis: {
        valueType: 'Category',
    },
    series:[{
        dataSource: chartData,
        xName: 'country', yName: 'gold',
        // Series type as column series
        type: 'Column',
        dataLabel: {
            visible: true,
            format: 'n2'
        }
    }],
    title: 'Olympic Medals',
    wallColor: 'transparent',
    enableRotation: true,
    rotation: 7,
    tilt: 10,
    depth: 100
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>

```



```

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Value	Format	Resultant Value	Description
1000	n1	1000.0	The number is rounded to 1 decimal place.
1000	n2	1000.00	The number is rounded to 2 decimal places.
1000	n3	1000.000	The number is rounded to 3 decimal place.
0.01	p1	1.0%	The number is converted to percentage with 1 decimal place.
0.01	p2	1.00%	The number is converted to percentage with 2 decimal place.
0.01	p3	1.000%	The number is converted to percentage with 3 decimal place.
1000	c1	\$1000.0	The currency symbol is appended to number and number is rounded to 1 decimal place.
1000	c2	\$1000.00	The currency symbol is appended to number and number is rounded to 2 decimal place.

Margin

The [margin](#) for data label can be applied by using [left](#), [right](#), [bottom](#) and [top](#) properties.

INDEX.JS

```

var chartData = [
    { country: "USA", gold: 50, silver: 70, bronze: 45 },
    { country: "China", gold: 40, silver: 60, bronze: 55 },
    { country: "Japan", gold: 70, silver: 60, bronze: 50 },

```

```

    { country: "Australia", gold: 60, silver: 56, bronze: 40 }
  ];
  var chart3D = new ej.charts.Chart3D({
    primaryXAxis: {
      valueType: 'Category',
    },
    series:[{
      dataSource: chartData,
      xName: 'country', yName: 'gold',
      type: 'Column',
      dataLabel: {
        visible: true,
        border:{
          width: 1,
          color : 'red'
        },
        margin:{
          left:5,
          right:5,
          top:5,
          bottom:5
        }
      }
    }
  ]},
  title: 'Olympic Medals',
  wallColor: 'transparent',
  enableRotation: true,
  rotation: 7,
  tilt: 10,
  depth: 100
}, '#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
```

```

    ele.style.visibility = "visible";
  }
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customization

The **stroke** and **border** of data label can be customized using [fill](#) and [border](#) properties.

INDEX.JS

```

var chartData = [
  { country: "USA", gold: 50, silver: 70, bronze: 45 },
  { country: "China", gold: 40, silver: 60, bronze: 55 },
  { country: "Japan", gold: 70, silver: 60, bronze: 50 },
  { country: "Australia", gold: 60, silver: 56, bronze: 40 }
];
var chart3D = new ej.charts.Chart3D({
  primaryXAxis: {
    valueType: 'Category',
  },
  series:[{
    dataSource: chartData,
    xName: 'country', yName: 'gold',
    type: 'Column',
    dataLabel: {
      visible: true,
      border:{ width: 2, color : 'red'},
    }
  }],
  title: 'Olympic Medals',
  wallColor: 'transparent',
  enableRotation: true,
  rotation: 7,
  tilt: 10,
  depth: 100
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

```

```

    <div id="container">
      <div id="element"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customizing specific label

A specific label can be customized by using the [textRender](#) event. The `textRender` event allows you to change the label text for the point.

INDEX.JS

```

var chartData = [
  { country: "USA", gold: 50, silver: 70, bronze: 45 },
  { country: "China", gold: 40, silver: 60, bronze: 55 },
  { country: "Japan", gold: 70, silver: 60, bronze: 50 },
  { country: "Australia", gold: 60, silver: 56, bronze: 40 }
];
var chart3D = new ej.charts.Chart3D({
  textRender: (args) => {
    if (args.point.index === 2) {
      args.text = 'Label';
    }
    else {
      args.cancel = true;
    }
  },
  primaryXAxis: {
    valueType: 'Category',
  },
  series:[{
    dataSource: [{ x: 'USA', y: 46 }, { x: 'GBR', y: 27 }, { x: 'CHN',
y: 26 }],
    xName: 'country', yName: 'gold',
    type: 'Column',
    dataLabel: {
      visible: true
    }
  }],
  title: 'Olympic Medals',
  wallColor: 'transparent',
  enableRotation: true,
  rotation: 7,
  tilt: 10,
  depth: 100
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

<!-- markdownlint-disable MD036 -->

Legend in EJ2 JavaScript 3D Chart control

<!-- markdownlint-disable MD036 -->

Legend provides information about the series rendered in the 3D chart.

Position and alignment

By using the [position](#) property, the legend can be positioned at left, right, top or bottom of the 3D chart. The legend is positioned at the bottom of the 3D chart, by default.

INDEX.JS

```

var chartData = [
  { country: "USA", gold: 50, silver: 70, bronze: 45 },
  { country: "China", gold: 40, silver: 60, bronze: 55 },
  { country: "Japan", gold: 70, silver: 60, bronze: 50 },
  { country: "Australia", gold: 60, silver: 56, bronze: 40 },
  { country: "France", gold: 50, silver: 45, bronze: 35 },
  { country: "Germany", gold: 40, silver: 30, bronze: 22 },
  { country: "Italy", gold: 40, silver: 35, bronze: 37 },
  { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart3D = new ej.charts.Chart3D({
  primaryXAxis: {
    valueType: 'Category',
    title: 'Countries'
  },

```

```

primaryYAxis: {
    minimum: 0, maximum: 80,
    interval: 20, title: 'Medals'
},
series:[{
    dataSource: chartData,
    xName: 'country', yName: 'gold',
    name: 'Gold', type: 'Column'
}, {
    dataSource: chartData,
    xName: 'country', yName: 'silver',
    name: 'Silver', type: 'Column'
}, {
    dataSource: chartData,
    xName: 'country', yName: 'bronze',
    name: 'Bronze', type: 'Column'
}],
title: 'Olympic Medals',
legendSettings: {
    visible: true,
    //Legend position as top
    position:'Top'
},
wallColor: 'transparent',
enableRotation: true,
rotation: 7,
tilt: 10,
depth: 100,
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}

```

```

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The custom position helps you to position the legend anywhere in the 3D chart using x and y coordinates.

INDEX.JS

```

var chartData = [
  { country: "USA", gold: 50, silver: 70, bronze: 45 },
  { country: "China", gold: 40, silver: 60, bronze: 55 },
  { country: "Japan", gold: 70, silver: 60, bronze: 50 },
  { country: "Australia", gold: 60, silver: 56, bronze: 40 },
  { country: "France", gold: 50, silver: 45, bronze: 35 },
  { country: "Germany", gold: 40, silver: 30, bronze: 22 },
  { country: "Italy", gold: 40, silver: 35, bronze: 37 },
  { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart3D = new ej.charts.Chart3D({
  primaryXAxis: {
    valueType: 'Category',
    title: 'Countries'
  },
  primaryYAxis: {
    minimum: 0, maximum: 80,
    interval: 20, title: 'Medals'
  },
  series:[{
    dataSource: chartData,
    xName: 'country', yName: 'gold',
    name: 'Gold', type: 'Column'
  }, {
    dataSource: chartData,
    xName: 'country', yName: 'silver',
    name: 'Silver', type: 'Column'
  }, {
    dataSource: chartData,
    xName: 'country', yName: 'bronze',
    name: 'Bronze', type: 'Column'
  }],
  title: 'Olympic Medals',
  legendSettings: {
    visible: true,
    position: 'Custom',
    location: { x: 200, y: 20 }
  },
  wallColor: 'transparent',
  enableRotation: true,
  rotation: 7,
  tilt: 10,
  depth: 100,
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Legend reverse

The order of the legend items can be reversed by using the [reverse](#) property. By default, legend for the first series in the collection will be placed first.

INDEX.JS

```

var chartData = [
  { country: "USA", gold: 50, silver: 70, bronze: 45 },
  { country: "China", gold: 40, silver: 60, bronze: 55 },
  { country: "Japan", gold: 70, silver: 60, bronze: 50 },
  { country: "Australia", gold: 60, silver: 56, bronze: 40 },
  { country: "France", gold: 50, silver: 45, bronze: 35 },
  { country: "Germany", gold: 40, silver: 30, bronze: 22 },
  { country: "Italy", gold: 40, silver: 35, bronze: 37 },
  { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart3D = new ej.charts.Chart3D({
  primaryXAxis: {
    valueType: 'Category',
    title: 'Countries'
  },
  primaryYAxis: {
    minimum: 0, maximum: 80,
    interval: 20, title: 'Medals'
  },
  series:[{
    dataSource: chartData,
    xName: 'country', yName: 'gold',

```



```

        name: 'Gold', type: 'Column'
    }, {
        dataSource: chartData,
        xName: 'country', yName: 'silver',
        name: 'Silver', type: 'Column'
    }, {
        dataSource: chartData,
        xName: 'country', yName: 'bronze',
        name: 'Bronze', type: 'Column'
    }],
    title: 'Olympic Medals',
    legendSettings: {
        visible: true,
        reverse: true
    },
    wallColor: 'transparent',
    enableRotation: true,
    rotation: 7,
    tilt: 10,
    depth: 100,
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Legend alignment

The legend can be aligned at near, far or center to the 3D chart using the [alignment](#) property.

INDEX.JS

```

var chartData = [
  { country: "USA", gold: 50, silver: 70, bronze: 45 },
  { country: "China", gold: 40, silver: 60, bronze: 55 },
  { country: "Japan", gold: 70, silver: 60, bronze: 50 },
  { country: "Australia", gold: 60, silver: 56, bronze: 40 },
  { country: "France", gold: 50, silver: 45, bronze: 35 },
  { country: "Germany", gold: 40, silver: 30, bronze: 22 },
  { country: "Italy", gold: 40, silver: 35, bronze: 37 },
  { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart3D = new ej.charts.Chart3D({
  primaryXAxis: {
    valueType: 'Category',
    title: 'Countries'
  },
  primaryYAxis: {
    minimum: 0, maximum: 80,
    interval: 20, title: 'Medals'
  },
  series:[{
    dataSource: chartData,
    xName: 'country', yName: 'gold',
    name: 'Gold', type: 'Column'
  }, {
    dataSource: chartData,
    xName: 'country', yName: 'silver',
    name: 'Silver', type: 'Column'
  }, {
    dataSource: chartData,
    xName: 'country', yName: 'bronze',
    name: 'Bronze', type: 'Column'
  }],
  title: 'Olympic Medals',
  legendSettings: {
    visible: true,
    position: 'Top',
    alignment: 'Near'
  },
  wallColor: 'transparent',
  enableRotation: true,
  rotation: 7,
  tilt: 10,
  depth: 100,
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Legend customization

To change the legend icon shape, [legendShape](#) property in the [series](#) can be used. By default, the legend icon shape is [seriesType](#).

INDEX.JS

```

var chartData = [
    { country: "USA", gold: 50, silver: 70, bronze: 45 },
    { country: "China", gold: 40, silver: 60, bronze: 55 },
    { country: "Japan", gold: 70, silver: 60, bronze: 50 },
    { country: "Australia", gold: 60, silver: 56, bronze: 40 },
    { country: "France", gold: 50, silver: 45, bronze: 35 },
    { country: "Germany", gold: 40, silver: 30, bronze: 22 },
    { country: "Italy", gold: 40, silver: 35, bronze: 37 },
    { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart3D = new ej.charts.Chart3D({
    primaryXAxis: {
        valueType: 'Category',
        title: 'Countries'
    },
    primaryYAxis: {
        minimum: 0, maximum: 80,
        interval: 20, title: 'Medals'
    },
    series:[{
        dataSource: chartData,
        xName: 'country', yName: 'gold',
        name: 'Gold', type: 'Column',
        legendShape: 'Circle'
    }, {
        dataSource: chartData,
        xName: 'country', yName: 'silver',
        name: 'Silver', type: 'Column',
        legendShape: 'SeriesType'
    }
]

```

```

    }, {
      dataSource: chartData,
      xName: 'country', yName: 'bronze',
      name: 'Bronze', type: 'Column',
      legendShape: 'Rectangle'
    }],
    title: 'Olympic Medals',
    legendSettings: {
      visible: true
    },
    wallColor: 'transparent',
    enableRotation: true,
    rotation: 7,
    tilt: 10,
    depth: 100,
  }, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Legend size

By default, legend takes 20% - 25% of the 3D chart's height horizontally, when it is placed on top or bottom position and 20% - 25% of the 3D chart's width vertically, when it is placed on left or right position. You can change this default legend size by using the [height](#) and [width](#) properties of the `legendSettings`.

INDEX.JS

```

var chartData = [
  { country: "USA", gold: 50, silver: 70, bronze: 45 },
  { country: "China", gold: 40, silver: 60, bronze: 55 },
  { country: "Japan", gold: 70, silver: 60, bronze: 50 },
  { country: "Australia", gold: 60, silver: 56, bronze: 40 },
  { country: "France", gold: 50, silver: 45, bronze: 35 },
  { country: "Germany", gold: 40, silver: 30, bronze: 22 },
  { country: "Italy", gold: 40, silver: 35, bronze: 37 },
  { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart3D = new ej.charts.Chart3D({
  primaryXAxis: {
    valueType: 'Category',
    title: 'Countries'
  },
  primaryYAxis: {
    minimum: 0, maximum: 80,
    interval: 20, title: 'Medals'
  },
  series:[{
    dataSource: chartData,
    xName: 'country', yName: 'gold',
    name: 'Gold', type: 'Column',
    legendShape: 'Circle'
  }, {
    dataSource: chartData,
    xName: 'country', yName: 'silver',
    name: 'Silver', type: 'Column',
    legendShape: 'Circle'
  }, {
    dataSource: chartData,
    xName: 'country', yName: 'bronze',
    name: 'Bronze', type: 'Column',
    legendShape: 'Circle'
  }],
  title: 'Olympic Medals',
  legendSettings: {
    visible: true,
    width: '500', height: '100',
    border: { width: 1, color: 'pink' }
  },
  wallColor: 'transparent',
  enableRotation: true,
  rotation: 7,
  tilt: 10,
  depth: 100,
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">

```

```

<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Legend item size

The size of the legend items can be customised by using the [shapeHeight](#) and [shapeWidth](#) properties.

INDEX.JS

```

var chartData = [
    { country: "USA", gold: 50, silver: 70, bronze: 45 },
    { country: "China", gold: 40, silver: 60, bronze: 55 },
    { country: "Japan", gold: 70, silver: 60, bronze: 50 },
    { country: "Australia", gold: 60, silver: 56, bronze: 40 },
    { country: "France", gold: 50, silver: 45, bronze: 35 },
    { country: "Germany", gold: 40, silver: 30, bronze: 22 },
    { country: "Italy", gold: 40, silver: 35, bronze: 37 },
    { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart3D = new ej.charts.Chart3D({
    primaryXAxis: {
        valueType: 'Category',
        title: 'Countries'
    },
    primaryYAxis: {
        minimum: 0, maximum: 80,
        interval: 20, title: 'Medals'
    },
    series:[{
        dataSource: chartData,
        xName: 'country', yName: 'gold',
        name: 'Gold', type: 'Column',
        legendShape: 'Circle'
    }, {
        dataSource: chartData,
        xName: 'country', yName: 'silver',
        name: 'Silver', type: 'Column',
        legendShape: 'Circle'
    }
]);

```

```

    }, {
      dataSource: chartData,
      xName: 'country', yName: 'bronze',
      name: 'Bronze', type: 'Column',
      legendShape: 'Circle'
    }],
    title: 'Olympic Medals',
    legendSettings: {
      visible: true,
      shapeHeight: 10, shapeWidth: 10
    },
    wallColor: 'transparent',
    enableRotation: true,
    rotation: 7,
    tilt: 10,
    depth: 100,
  }, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Paging for legend

Paging will be enabled by default, when the legend items exceeds the legend bounds. Each legend items can be viewed by navigating between the pages using navigation buttons.

INDEX.JS

```

var chartData = [

```

```

    { x: 'WW', y: 12, y1: 22, y2: 38.3, y3: 50 },
    { x: 'EU', y: 9.9, y1: 26, y2: 45.2, y3: 63.6 },
    { x: 'APAC', y: 4.4, y1: 9.3, y2: 18.2, y3: 20.9 },
    { x: 'LATAM', y: 6.4, y1: 28, y2: 46.7, y3: 65.1 },
    { x: 'MEA', y: 30, y1: 45.7, y2: 61.5, y3: 73 },
    { x: 'NA', y: 25.3, y1: 35.9, y2: 64, y3: 81.4 }
  ];
  var chart3D = new ej.charts.Chart3D({
    primaryXAxis: {
      title: 'Countries',
      valueType: 'Category', interval: 1,
      labelIntersectAction : 'Rotate45'
    },
    primaryYAxis:
    {
      title: 'Penetration (%)',
      labelFormat: '{value}%',
      minimum: 0, maximum: 90
    },
    series: [
      {
        type: 'Column', name: 'December 2007',
        dataSource: chartData, xName: 'x', yName: 'y'
      }, {
        type: 'Column', name: 'December 2008',
        dataSource: chartData, xName: 'x', yName: 'y1'
      }, {
        type: 'Column', name: 'December 2009',
        dataSource: chartData, xName: 'x', yName: 'y2'
      }, {
        type: 'Column', name: 'December 2010',
        dataSource: chartData, xName: 'x', yName: 'y3'
      }
    ],
    title: 'FB Penetration of Internet Audience',
    legendSettings: {
      padding: 10, shapePadding: 10,
      visible: true, border: {
        width: 2, color: 'grey'
      },
      width: '200'
    },
    wallColor: 'transparent',
    enableRotation: true,
    rotation: 7,
    tilt: 10,
    depth: 100,
  }, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">

```



```

<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Legend text wrap

When the legend text exceeds the container, the text can be wrapped by using the [textWrap](#) property. End user can also wrap the legend text based on the [maximumLabelWidth](#) property.

INDEX.JS

```

var chartData = [
    { country: "USA", gold: 50, silver: 70, bronze: 45 },
    { country: "China", gold: 40, silver: 60, bronze: 55 },
    { country: "Japan", gold: 70, silver: 60, bronze: 50 },
    { country: "Australia", gold: 60, silver: 56, bronze: 40 },
    { country: "France", gold: 50, silver: 45, bronze: 35 },
    { country: "Germany", gold: 40, silver: 30, bronze: 22 },
    { country: "Italy", gold: 40, silver: 35, bronze: 37 },
    { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart3D = new ej.charts.Chart3D({
    primaryXAxis: {
        valueType: 'Category',
        title: 'Countries'
    },
    primaryYAxis: {
        minimum: 0, maximum: 80,
        interval: 20, title: 'Medals'
    },
    series:[{
        dataSource: chartData,
        xName: 'country', yName: 'gold',
        name: 'Gold', type: 'Column'
    }, {
        dataSource: chartData,
        xName: 'country', yName: 'silver',
        name: 'Silver', type: 'Column'
    }
]

```

```

    }, {
      dataSource: chartData,
      xName: 'country', yName: 'bronze',
      name: 'Bronze', type: 'Column'
    }],
    title: 'Olympic Medals',
    legendSettings: {
      visible: true,
      position: 'Right',
      textWrap: 'Wrap',
      maximumLabelWidth: 50,
    },
    wallColor: 'transparent',
    enableRotation: true,
    rotation: 7,
    tilt: 10,
    depth: 100
  }, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Series selection through legend

By default, you can collapse the series visibility by clicking the legend. On the other hand, turn off the [toggleVisibility](#) property if you must use a legend click to choose a series.

INDEX.JS

```

var chartData = [
  { country: "USA", gold: 50, silver: 70, bronze: 45 },
  { country: "China", gold: 40, silver: 60, bronze: 55 },
  { country: "Japan", gold: 70, silver: 60, bronze: 50 },
  { country: "Australia", gold: 60, silver: 56, bronze: 40 },
  { country: "France", gold: 50, silver: 45, bronze: 35 },
  { country: "Germany", gold: 40, silver: 30, bronze: 22 },
  { country: "Italy", gold: 40, silver: 35, bronze: 37 },
  { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart3D = new ej.charts.Chart3D({
  primaryXAxis: {
    valueType: 'Category',
    title: 'Countries'
  },
  primaryYAxis: {
    minimum: 0, maximum: 80,
    interval: 20, title: 'Medals'
  },
  series:[{
    dataSource: chartData,
    xName: 'country', yName: 'gold',
    name: 'Gold', type: 'Column'
  }, {
    dataSource: chartData,
    xName: 'country', yName: 'silver',
    name: 'Silver', type: 'Column'
  }, {
    dataSource: chartData,
    xName: 'country', yName: 'bronze',
    name: 'Bronze', type: 'Column'
  }],
  selectionMode: 'Series',
  legendSettings: {
    visible: true,
    toggleVisibility: false
  },
  title: 'Olympic Medals',
  wallColor: 'transparent',
  enableRotation: true,
  rotation: 7,
  tilt: 10,
  depth: 100
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Collapsing legend item

By default, series name will be displayed as legend. To skip the legend for a particular series, you can give empty string to the series name.

INDEX.JS

```

var chartData = [
    { country: "USA", gold: 50, silver: 70, bronze: 45 },
    { country: "China", gold: 40, silver: 60, bronze: 55 },
    { country: "Japan", gold: 70, silver: 60, bronze: 50 },
    { country: "Australia", gold: 60, silver: 56, bronze: 40 },
    { country: "France", gold: 50, silver: 45, bronze: 35 },
    { country: "Germany", gold: 40, silver: 30, bronze: 22 },
    { country: "Italy", gold: 40, silver: 35, bronze: 37 },
    { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart3D = new ej.charts.Chart3D({
    primaryXAxis: {
        valueType: 'Category',
        title: 'Countries'
    },
    primaryYAxis: {
        minimum: 0, maximum: 80,
        interval: 20, title: 'Medals'
    },
    series:[{
        dataSource: chartData,
        xName: 'country', yName: 'gold',
        name: 'Gold', type: 'Column'
    }, {
        dataSource: chartData,
        xName: 'country', yName: 'silver',
        name: 'Silver', type: 'Column'
    }, {
        dataSource: chartData,
        xName: 'country', yName: 'bronze',
        name: 'Bronze', type: 'Column'
    }
]

```

```

    }],
    legendSettings: {
        visible: true,
        toggleVisibility: true
    },
    title: 'Olympic Medals',
    wallColor: 'transparent',
    enableRotation: true,
    rotation: 7,
    tilt: 10,
    depth: 100
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Legend title

You can set title for legend using [title](#) property in [legendSettings](#). The [size](#), [color](#), [opacity](#), [fontStyle](#), [fontWeight](#), [fontFamily](#), [textAlignment](#), and [textOverflow](#) of legend title can be customized by using the [titleStyle](#) property in [legendSettings](#). The [titlePosition](#) is used to set the legend position in Top, Left and Right position. The [maximumTitleWidth](#) is used to set the width of the legend title. By default, it will be 100px.

INDEX.JS

```

var chartData = [
  { country: "USA", gold: 50, silver: 70, bronze: 45 },
  { country: "China", gold: 40, silver: 60, bronze: 55 },

```

```

    { country: "Japan", gold: 70, silver: 60, bronze: 50 },
    { country: "Australia", gold: 60, silver: 56, bronze: 40 },
    { country: "France", gold: 50, silver: 45, bronze: 35 },
    { country: "Germany", gold: 40, silver: 30, bronze: 22 },
    { country: "Italy", gold: 40, silver: 35, bronze: 37 },
    { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
  ];
  var chart3D = new ej.charts.Chart3D({
    primaryXAxis: {
      valueType: 'Category'
    },
    primaryYAxis: {
      minimum: 0, maximum: 80,
      interval: 20, title: 'Medals'
    },
    series:[{
      dataSource: chartData,
      xName: 'country', yName: 'gold',
      name: 'Gold', type: 'Column'
    }, {
      dataSource: chartData,
      xName: 'country', yName: 'silver',
      name: 'Silver', type: 'Column'
    }, {
      dataSource: chartData,
      xName: 'country', yName: 'bronze',
      name: 'Bronze', type: 'Column'
    }],
    legendSettings: {
      visible: true,
      title: 'Countries'
    },
    title: 'Olympic Medals',
    wallColor: 'transparent',
    enableRotation: true,
    rotation: 7,
    tilt: 10,
    depth: 100
  }, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>

```

```

<body>

    <div id="container">
        <div id="element"></div>
    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Arrow page navigation

The page number will always be visible while using legend paging. It is now possible to disable the page number and enable page navigation with the left and right arrows. The [enablePages](#) property needs to be set to **false** in order to render the arrow page navigation.

INDEX.JS

```

var chartData = [
    { x: 'WW', y: 12, y1: 22, y2: 38.3, y3: 50 },
    { x: 'EU', y: 9.9, y1: 26, y2: 45.2, y3: 63.6 },
    { x: 'APAC', y: 4.4, y1: 9.3, y2: 18.2, y3: 20.9 },
    { x: 'LATAM', y: 6.4, y1: 28, y2: 46.7, y3: 65.1 },
    { x: 'MEA', y: 30, y1: 45.7, y2: 61.5, y3: 73 },
    { x: 'NA', y: 25.3, y1: 35.9, y2: 64, y3: 81.4 }
];
var chart3D = new ej.charts.Chart3D({
    primaryXAxis: {
        title: 'Countries',
        valueType: 'Category', interval: 1,
        labelIntersectAction : 'Rotate45'
    },
    primaryYAxis:
    {
        title: 'Penetration (%)',
        labelFormat: '{value}%',
        minimum: 0, maximum: 90
    },
    series: [
        {
            type: 'Column', name: 'December 2007',
            dataSource: chartData, xName: 'x', yName: 'y'
        }, {
            type: 'Column', name: 'December 2008',
            dataSource: chartData, xName: 'x', yName: 'y1'
        }, {
            type: 'Column', name: 'December 2009',
            dataSource: chartData, xName: 'x', yName: 'y2'
        }, {
            type: 'Column', name: 'December 2010',
            dataSource: chartData, xName: 'x', yName: 'y3'
        }
    ]
},

```

```

    title: 'FB Penetration of Internet Audience',
    legendSettings: {
        width: '180',
        height: '20',
        enablePages: false
    },
    wallColor: 'transparent',
    enableRotation: true,
    rotation: 7,
    tilt: 10,
    depth: 100
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Legend item padding

The [itemPadding](#) property can be used to adjust the space between the legend items.

INDEX.JS

```

var chartData = [
  { x: 'WW', y: 12, y1: 22, y2: 38.3, y3: 50 },
  { x: 'EU', y: 9.9, y1: 26, y2: 45.2, y3: 63.6 },
  { x: 'APAC', y: 4.4, y1: 9.3, y2: 18.2, y3: 20.9 },
  { x: 'LATAM', y: 6.4, y1: 28, y2: 46.7, y3: 65.1 },
  { x: 'MEA', y: 30, y1: 45.7, y2: 61.5, y3: 73 },
  { x: 'NA', y: 25.3, y1: 35.9, y2: 64, y3: 81.4 }
];

```



```

var chart3D = new ej.charts.Chart3D({
  primaryXAxis: {
    title: 'Countries',
    valueType: 'Category', interval: 1,
    labelIntersectAction : 'Rotate45'
  },
  primaryYAxis:
  {
    title: 'Penetration (%)',
    labelFormat: '{value}%',
    minimum: 0, maximum: 90
  },
  series: [
    {
      type: 'Column', name: 'December 2007',
      dataSource: chartData, xName: 'x', yName: 'y'
    }, {
      type: 'Column', name: 'December 2008',
      dataSource: chartData, xName: 'x', yName: 'y1'
    }, {
      type: 'Column', name: 'December 2009',
      dataSource: chartData, xName: 'x', yName: 'y2'
    }, {
      type: 'Column', name: 'December 2010',
      dataSource: chartData, xName: 'x', yName: 'y3'
    }
  ],
  title: 'FB Penetration of Internet Audience',
  legendSettings: {
    enablePages: false,
    itemPadding: 30
  },
  wallColor: 'transparent',
  enableRotation: true,
  rotation: 7,
  tilt: 10,
  depth: 100
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

```

```

    <div id="container">
      <div id="element"></div>
    </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Note: To use the legend feature, we need to inject the **Legend3D** module using **Chart3D.Inject(Legend3D)** method.

Tooltip in EJ2 JavaScript 3D Chart control

<!-- markdownlint-disable MD036 -->

The 3D Chart will display details about the points through tooltip, when the mouse is moved over the specific point.

Default tooltip

By default, tooltip is not visible. The tooltip can be enabled by setting the **enable** property in **tooltipSettings** to **true** and by injecting **Tooltip3D** module using **Chart3D.Inject(Tooltip3D)**.

INDEX.JS

```

var chartData = [
  { month: 'Jan', sales: 35 }, { month: 'Feb', sales: 28 },
  { month: 'Mar', sales: 34 }, { month: 'Apr', sales: 32 },
  { month: 'May', sales: 40 }, { month: 'Jun', sales: 32 },
  { month: 'Jul', sales: 35 }, { month: 'Aug', sales: 55 },
  { month: 'Sep', sales: 38 }, { month: 'Oct', sales: 30 },
  { month: 'Nov', sales: 25 }, { month: 'Dec', sales: 32 }
];
var chart3D = new ej.charts.Chart3D({
  tooltip: { enable: true },
  primaryXAxis: {
    valueType: 'Category',
    labelRotation: -45,
    labelPlacement: 'BetweenTicks'
  },
  series:[{
    dataSource: chartData,
    xName: 'month',
    yName: 'sales',
    type: 'Column'
  }],
  wallColor: 'transparent',
  enableRotation: true,
  rotation: 7,
  tilt: 10,
  depth: 100,
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

<!-- markdownlint-disable MD013 -->

Fixed tooltip

By default, tooltip track the mouse movement, but the tooltip can be set in fixed position by using the [location](#) property.

INDEX.JS

```

var chartData = [
  { month: 'Jan', sales: 35 }, { month: 'Feb', sales: 28 },
  { month: 'Mar', sales: 34 }, { month: 'Apr', sales: 32 },
  { month: 'May', sales: 40 }, { month: 'Jun', sales: 32 },
  { month: 'Jul', sales: 35 }, { month: 'Aug', sales: 55 },
  { month: 'Sep', sales: 38 }, { month: 'Oct', sales: 30 },
  { month: 'Nov', sales: 25 }, { month: 'Dec', sales: 32 }
];
var chart3D = new ej.charts.Chart3D({
  tooltip: {
    enable: true,
    location: { x: 120, y: 20 }
  },
  primaryXAxis: {
    valueType: 'Category',
    labelRotation: -45,
    labelPlacement: 'BetweenTicks'
  }
});

```

```

    },
    series:[{
        dataSource: chartData,
        xName: 'month',
        yName: 'sales',
        type: 'Column'
    }],
    wallColor: 'transparent',
    enableRotation: true,
    rotation: 7,
    tilt: 10,
    depth: 100,
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Format the tooltip

<!-- markdownlint-disable MD013 -->

By default, tooltip shows information of x and y value in points. In addition to that, more information can be shown in tooltip by using the [format](#) property. For example the format `${series.name}` : `${point.y}` shows series name and point y value.

INDEX.JS

```

var chartData = [
  { month: 'Jan', sales: 35 }, { month: 'Feb', sales: 28 },

```

```

    { month: 'Mar', sales: 34 }, { month: 'Apr', sales: 32 },
    { month: 'May', sales: 40 }, { month: 'Jun', sales: 32 },
    { month: 'Jul', sales: 35 }, { month: 'Aug', sales: 55 },
    { month: 'Sep', sales: 38 }, { month: 'Oct', sales: 30 },
    { month: 'Nov', sales: 25 }, { month: 'Dec', sales: 32 }
  ];
  var chart3D = new ej.charts.Chart3D({
    tooltip: {
      enable: true,
      header: 'Unemployment',
      format: '<b>${series.name} : ${point.y}</b>'
    },
    primaryXAxis: {
      valueType: 'Category',
      labelRotation: -45,
      labelPlacement: 'BetweenTicks'
    },
    series:[{
      dataSource: chartData,
      xName: 'month',
      yName: 'sales',
      type: 'Column',
      name: "Month"
    }],
    wallColor: 'transparent',
    enableRotation: true,
    rotation: 7,
    tilt: 10,
    depth: 100
  }, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";

```

```

}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Tooltip template

Any HTML elements can be displayed in the tooltip by using the [template](#) property of the tooltip. The `{x}` and `{y}` can be used as place holders in the HTML element to display the x and y values of the corresponding data point.

INDEX.JS

```

var chartData = [
  { month: 'Jan', sales: 35 }, { month: 'Feb', sales: 28 },
  { month: 'Mar', sales: 34 }, { month: 'Apr', sales: 32 },
  { month: 'May', sales: 40 }, { month: 'Jun', sales: 32 },
  { month: 'Jul', sales: 35 }, { month: 'Aug', sales: 55 },
  { month: 'Sep', sales: 38 }, { month: 'Oct', sales: 30 },
  { month: 'Nov', sales: 25 }, { month: 'Dec', sales: 32 }
];
var chart3D = new ej.charts.Chart3D({
  tooltip: {
    enable: true,
    template: '#Unemployment'
  },
  primaryXAxis: {
    valueType: 'Category',
    labelRotation: -45,
    labelPlacement: 'BetweenTicks'
  },
  series:[{
    dataSource: chartData,
    xName: 'month',
    yName: 'sales',
    type: 'Column'
  }],
  wallColor: 'transparent',
  enableRotation: true,
  rotation: 7,
  tilt: 10,
  depth: 100,
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
        <script id="Unemployment" type="text/x-template">
            <div id='templateWrap'>
                <table style="width:100%; border: 1px solid black;">
                    <tr><th colspan="2" bgcolor="#00FFFF">Unemployment</th></tr>
                    <tr><td bgcolor="#00FFFF">${x}</td><td
bgcolor="#00FFFF">${y}</td></tr>
                </table>
            </div>
        </script>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customize the appearance of tooltip

The [fill](#) and [border](#) properties are used to customize the background color and border of the tooltip respectively. The [textStyle](#) property in the tooltip is used to customize the font of the tooltip text.

INDEX.JS

```

var chartData = [
    { month: 'Jan', sales: 35 }, { month: 'Feb', sales: 28 },
    { month: 'Mar', sales: 34 }, { month: 'Apr', sales: 32 },
    { month: 'May', sales: 40 }, { month: 'Jun', sales: 32 },
    { month: 'Jul', sales: 35 }, { month: 'Aug', sales: 55 },
    { month: 'Sep', sales: 38 }, { month: 'Oct', sales: 30 },
    { month: 'Nov', sales: 25 }, { month: 'Dec', sales: 32 }
];
var chart3D = new ej.charts.Chart3D({
    tooltip: {
        enable: true,
        format: '${series.name} ${point.x} : ${point.y}',
        fill: '#7bb4eb',
        border: {
            width: 2,
            color: 'grey'
        }
    },
    primaryXAxis: {
        valueType: 'Category',
        labelRotation: -45,
        labelPlacement: 'BetweenTicks'
    }
});

```

```

    },
    series:[{
        dataSource: chartData,
        xName: 'month',
        yName: 'sales',
        type: 'Column',
        name: 'China',
    }],
    wallColor: 'transparent',
    enableRotation: true,
    rotation: 7,
    tilt: 10,
    depth: 100,
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

<!-- markdownlint-disable MD036 -->

Selection in EJ2 JavaScript 3D Chart control

The 3D chart provides selection support for the series and its data points on mouse click.

When mouse is clicked on the data points, the corresponding series legend will also be selected.

We have different types of selection mode for selecting a data.

- None

- Point
- Series
- Cluster

Point

To select a point, set the [selectionMode](#) property to **Point**.

INDEX.JS

```
var chartData = [
  { country: "USA", gold: 50, silver: 70, bronze: 45 },
  { country: "China", gold: 40, silver: 60, bronze: 55 },
  { country: "Japan", gold: 70, silver: 60, bronze: 50 },
  { country: "Australia", gold: 60, silver: 56, bronze: 40 },
  { country: "France", gold: 50, silver: 45, bronze: 35 },
  { country: "Germany", gold: 40, silver: 30, bronze: 22 },
  { country: "Italy", gold: 40, silver: 35, bronze: 37 },
  { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart3D = new ej.charts.Chart3D({
  primaryXAxis: {
    valueType: 'Category',
  },
  series:[{
    dataSource: chartData,
    xName: 'country', yName: 'gold',
    name: 'Gold', type: 'Column'
  }, {
    dataSource: chartData,
    xName: 'country', yName: 'silver',
    name: 'Silver', type: 'Column'
  }, {
    dataSource: chartData,
    xName: 'country', yName: 'bronze',
    name: 'Bronze', type: 'Column'
  }],
  title: 'Olympic Medals',
  selectionMode: 'Point',
  wallColor: 'transparent',
  enableRotation: true,
  rotation: 7,
  tilt: 10,
  depth: 100
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Series

To select a series, set the [selectionMode](#) property to **Series**.

INDEX.JS

```

var chartData = [
    { country: "USA", gold: 50, silver: 70, bronze: 45 },
    { country: "China", gold: 40, silver: 60, bronze: 55 },
    { country: "Japan", gold: 70, silver: 60, bronze: 50 },
    { country: "Australia", gold: 60, silver: 56, bronze: 40 },
    { country: "France", gold: 50, silver: 45, bronze: 35 },
    { country: "Germany", gold: 40, silver: 30, bronze: 22 },
    { country: "Italy", gold: 40, silver: 35, bronze: 37 },
    { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart3D = new ej.charts.Chart3D({
    primaryXAxis: {
        valueType: 'Category',
    },
    series:[{
        dataSource: chartData,
        xName: 'country', yName: 'gold',
        name: 'Gold', type: 'Column'
    }, {
        dataSource: chartData,
        xName: 'country', yName: 'silver',
        name: 'Silver', type: 'Column'
    }, {
        dataSource: chartData,
        xName: 'country', yName: 'bronze',
        name: 'Bronze', type: 'Column'
    }],
    title: 'Olympic Medals',
    selectionMode: 'Series',
    wallColor: 'transparent',
    enableRotation: true,
    rotation: 7,

```

```

    tilt: 10,
    depth: 100
  }, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Cluster

To select the points that corresponds to the same index in all the series, set the [selectionMode](#) property to **Cluster**.

INDEX.JS

```

var chartData = [
  { country: "USA", gold: 50, silver: 70, bronze: 45 },
  { country: "China", gold: 40, silver: 60, bronze: 55 },
  { country: "Japan", gold: 70, silver: 60, bronze: 50 },
  { country: "Australia", gold: 60, silver: 56, bronze: 40 },
  { country: "France", gold: 50, silver: 45, bronze: 35 },
  { country: "Germany", gold: 40, silver: 30, bronze: 22 },
  { country: "Italy", gold: 40, silver: 35, bronze: 37 },
  { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart3D = new ej.charts.Chart3D({
  primaryXAxis: {
    valueType: 'Category',
  },
  series:[{

```

```

        dataSource: chartData,
        xName: 'country', yName: 'gold',
        name: 'Gold', type: 'Column'
    }, {
        dataSource: chartData,
        xName: 'country', yName: 'silver',
        name: 'Silver', type: 'Column'
    }, {
        dataSource: chartData,
        xName: 'country', yName: 'bronze',
        name: 'Bronze', type: 'Column'
    }
  ],
  title: 'Olympic Medals',
  selectionMode: 'Cluster',
  wallColor: 'transparent',
  enableRotation: true,
  rotation: 7,
  tilt: 10,
  depth: 100
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Selection type

To select multiple points or series, enable the [isMultiSelect](#) property.

INDEX.JS

```

var chartData = [
  { country: "USA", gold: 50, silver: 70, bronze: 45 },
  { country: "China", gold: 40, silver: 60, bronze: 55 },
  { country: "Japan", gold: 70, silver: 60, bronze: 50 },
  { country: "Australia", gold: 60, silver: 56, bronze: 40 },
  { country: "France", gold: 50, silver: 45, bronze: 35 },
  { country: "Germany", gold: 40, silver: 30, bronze: 22 },
  { country: "Italy", gold: 40, silver: 35, bronze: 37 },
  { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart3D = new ej.charts.Chart3D({
  primaryXAxis: {
    valueType: 'Category',
  },
  series:[{
    dataSource: chartData,
    xName: 'country', yName: 'gold',
    name: 'Gold', type: 'Column'
  }, {
    dataSource: chartData,
    xName: 'country', yName: 'silver',
    name: 'Silver', type: 'Column'
  }, {
    dataSource: chartData,
    xName: 'country', yName: 'bronze',
    name: 'Bronze', type: 'Column'
  }],
  title: 'Olympic Medals',
  selectionMode: 'Series',
  isMultiSelect: true,
  wallColor: 'transparent',
  enableRotation: true,
  rotation: 7,
  tilt: 10,
  depth: 100
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">

```

```

        <div id="element"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Selection during initial loading

In a 3D chart, selecting a point or series during initial loading can only be done programmatically. The [selectedDataIndexes](#) property can be used for this.

INDEX.JS

```

var chartData = [
    { country: "USA", gold: 50, silver: 70, bronze: 45 },
    { country: "China", gold: 40, silver: 60, bronze: 55 },
    { country: "Japan", gold: 70, silver: 60, bronze: 50 },
    { country: "Australia", gold: 60, silver: 56, bronze: 40 },
    { country: "France", gold: 50, silver: 45, bronze: 35 },
    { country: "Germany", gold: 40, silver: 30, bronze: 22 },
    { country: "Italy", gold: 40, silver: 35, bronze: 37 },
    { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart3D = new ej.charts.Chart3D({
    primaryXAxis: {
        valueType: 'Category',
    },
    series:[{
        dataSource: chartData,
        xName: 'country', yName: 'gold',
        name: 'Gold', type: 'Column'
    }, {
        dataSource: chartData,
        xName: 'country', yName: 'silver',
        name: 'Silver', type: 'Column'
    }, {
        dataSource: chartData,
        xName: 'country', yName: 'bronze',
        name: 'Bronze', type: 'Column'
    }],
    title: 'Olympic Medals',
    selectionMode: 'Point',
    isMultiSelect: true,
    selectedDataIndexes: [
        { series: 0, point: 1}, { series: 2, point: 3}
    ],
    wallColor: 'transparent',
    enableRotation: true,
    rotation: 7,
    tilt: 10,
    depth: 100
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Selection through legend

To select a point or series through legend use the [toggleVisibility](#) property. Also, use [enableHighlight](#) property for highlighting the series through legend.

INDEX.JS

```

var chartData = [
  { country: "USA", gold: 50, silver: 70, bronze: 45 },
  { country: "China", gold: 40, silver: 60, bronze: 55 },
  { country: "Japan", gold: 70, silver: 60, bronze: 50 },
  { country: "Australia", gold: 60, silver: 56, bronze: 40 },
  { country: "France", gold: 50, silver: 45, bronze: 35 },
  { country: "Germany", gold: 40, silver: 30, bronze: 22 },
  { country: "Italy", gold: 40, silver: 35, bronze: 37 },
  { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart3D = new ej.charts.Chart3D({
  primaryXAxis: {
    valueType: 'Category',
  },
  series:[{
    dataSource: chartData,
    xName: 'country', yName: 'gold',
    name: 'Gold', type: 'Column'
  }
]);

```

```

    }, {
      dataSource: chartData,
      xName: 'country', yName: 'silver',
      name: 'Silver', type: 'Column'
    }, {
      dataSource: chartData,
      xName: 'country', yName: 'bronze',
      name: 'Bronze', type: 'Column'
    }
  ],
  title: 'Olympic Medals',
  legendSettings: { visible: true, toggleVisibility: false,
enableHighlight: true},
  wallColor: 'transparent',
  enableRotation: true,
  rotation: 7,
  tilt: 10,
  depth: 100
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Print and Export in EJ2 JavaScript 3D Chart control

Print

The rendered 3D chart can be printed directly from the browser by calling the public method [print](#). The ID of the 3D chart's div element must be passed as the input parameter to that method.

INDEX.JS

```

var chartData = [
  { country: "USA", gold: 50, silver: 70, bronze: 45 },
  { country: "China", gold: 40, silver: 60, bronze: 55 },
  { country: "Japan", gold: 70, silver: 60, bronze: 50 },
  { country: "Australia", gold: 60, silver: 56, bronze: 40 },
  { country: "France", gold: 50, silver: 45, bronze: 35 },
  { country: "Germany", gold: 40, silver: 30, bronze: 22 },
  { country: "Italy", gold: 40, silver: 35, bronze: 37 },
  { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart3D = new ej.charts.Chart3D({
  primaryXAxis: {
    valueType: 'Category',
  },
  series:[{
    dataSource: chartData,
    xName: 'country', yName: 'gold',
    // Series type as column series
    type: 'Column'
  }],
  title: 'Olympic Medals',
  wallColor: 'transparent',
  enableRotation: true,
  rotation: 7,
  tilt: 10,
  depth: 100
}, '#element');
document.getElementById('print').onclick = () => {
  chart.print();
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
    <button id= "print" type="button" width ='15%' style="float:
right">Print</button>
  </div>

```

```
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Export

The rendered 3D chart can be exported to JPEG, PNG, SVG, or PDF format using the [export](#) method. The input parameters for this method are: `type` for format and `fileName` for result.

INDEX.JS

```
var chartData = [
  { country: "USA", gold: 50, silver: 70, bronze: 45 },
  { country: "China", gold: 40, silver: 60, bronze: 55 },
  { country: "Japan", gold: 70, silver: 60, bronze: 50 },
  { country: "Australia", gold: 60, silver: 56, bronze: 40 },
  { country: "France", gold: 50, silver: 45, bronze: 35 },
  { country: "Germany", gold: 40, silver: 30, bronze: 22 },
  { country: "Italy", gold: 40, silver: 35, bronze: 37 },
  { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart3D = new ej.charts.Chart3D({
  primaryXAxis: {
    valueType: 'Category',
  },
  series:[{
    dataSource: chartData,
    xName: 'country', yName: 'gold',
    // Series type as column series
    type: 'Column'
  }],
  title: 'Olympic Medals',
  wallColor: 'transparent',
  enableRotation: true,
  rotation: 7,
  tilt: 10,
  depth: 100
}, '#element');
document.getElementById('export').onclick = () => {
  chart.export('PNG', 'result');
};
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
        <button id= "export" type="button" width ='15%' style="float:
right">Export</button>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Appearance in EJ2 JavaScript 3D Chart control

Custom color palette

The default color of series or points can be customized by providing a custom color palette of your choice by using the [palettes](#) property.

INDEX.JS

```

var chartData = [
    { country: "USA", gold: 50, silver: 70, bronze: 45 },
    { country: "China", gold: 40, silver: 60, bronze: 55 },
    { country: "Japan", gold: 70, silver: 60, bronze: 50 },
    { country: "Australia", gold: 60, silver: 56, bronze: 40 },
    { country: "France", gold: 50, silver: 45, bronze: 35 },
    { country: "Germany", gold: 40, silver: 30, bronze: 22 },
    { country: "Italy", gold: 40, silver: 35, bronze: 37 },
    { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart3D = new ej.charts.Chart3D({
    primaryXAxis: {
        valueType: 'Category',
    },
    series:[{
        dataSource: chartData,
        xName: 'country', yName: 'gold',
        name: 'Gold', type: 'Column'
    }, {
        dataSource: chartData,
        xName: 'country', yName: 'silver',
        name: 'Silver', type: 'Column'
    }, {
        dataSource: chartData,
        xName: 'country', yName: 'bronze',

```

```

        name: 'Bronze', type: 'Column'
    }],
    title: 'Olympic Medals',
    wallColor: 'transparent',
    palettes: ["#E94649", "#F6B53F", "#6FAAB0", "#C4C24A"],
    enableRotation: true,
    rotation: 7,
    tilt: 10,
    depth: 100
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Data point customization

The color of an individual data point can be customized using the below options.

Point color mapping

The color for the points can be bound from the [dataSource](#) for the series by utilizing the [pointColorMapping](#) property.

INDEX.JS

```

var chart3D = new ej.charts.Chart3D({
  primaryXAxis: {
    valueType: 'Category',
  },
  series:[{

```

```

    pointColorMapping: "color",
    dataSource: [
      { x: 'Jan', y: 6.96, color: "#ed4c40" },
      { x: 'Feb', y: 8.9, color: "#3285f3"},
      { x: 'Mar', y: 12, color: "#1dd7f3"},
      { x: 'Apr', y: 17.5, color: "#fe1684" },
      { x: 'May', y: 22.1, color: "#4633f2" }
    ], xName: 'x', yName: 'y', type: 'Column',
  }],
  title: 'Olympic Medals',
  wallColor: 'transparent',
  enableRotation: true,
  rotation: 7,
  tilt: 10,
  depth: 100
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Point level customization

The data label and fill color of each data point can be customized using the [pointRender](#) and [textRender](#) events.

INDEX.JS

```

var chartData = [
  { country: "USA", gold: 50, silver: 70, bronze: 45 },

```

```

    { country: "China", gold: 40, silver: 60, bronze: 55 },
    { country: "Japan", gold: 70, silver: 60, bronze: 50 },
    { country: "Australia", gold: 60, silver: 56, bronze: 40 },
    { country: "France", gold: 50, silver: 45, bronze: 35 },
    { country: "Germany", gold: 40, silver: 30, bronze: 22 },
    { country: "Italy", gold: 40, silver: 35, bronze: 37 },
    { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
  ];
  var colors = ['#00bdae', '#404041', '#357cd2', '#e56590', '#f8b883',
    '#70ad47', '#dd8abd', '#7f84e8', '#7bb4eb', '#ea7a57'];
  var chart3D = new ej.charts.Chart3D({
    pointRender: (args) => {
      args.fill = colors[args.point.index];
    },
    primaryXAxis: {
      valueType: 'Category',
    },
    series:[{
      dataSource: chartData,
      xName: 'country', yName: 'gold', type: 'Column',
    }],
    title: 'Olympic Medals',
    wallColor: 'transparent',
    enableRotation: true,
    rotation: 7,
    tilt: 10,
    depth: 100
  }, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
  var ele = document.getElementById('container');
  if(ele) {
    ele.style.visibility = "visible";
  }

  </script>

```

```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

```
<!-- markdownlint-disable MD036 -->
```

Chart area customization

```
<!-- markdownlint-disable MD036 -->
```

Customize the chart background

The background color and border of the 3D chart can be customized using the [background](#) and [border](#) properties.

INDEX.JS

```
var chartData = [
  { country: "USA", gold: 50, silver: 70, bronze: 45 },
  { country: "China", gold: 40, silver: 60, bronze: 55 },
  { country: "Japan", gold: 70, silver: 60, bronze: 50 },
  { country: "Australia", gold: 60, silver: 56, bronze: 40 },
  { country: "France", gold: 50, silver: 45, bronze: 35 },
  { country: "Germany", gold: 40, silver: 30, bronze: 22 },
  { country: "Italy", gold: 40, silver: 35, bronze: 37 },
  { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart3D = new ej.charts.Chart3D({
  primaryXAxis: {
    valueType: 'Category',
  },
  series:[{
    dataSource: chartData,
    xName: 'country', yName: 'gold', type: 'Column',
  }],
  title: 'Olympic Medals',
  wallColor: 'transparent',
  enableRotation: true,
  rotation: 7,
  tilt: 10,
  depth: 100,
  background: 'skyblue',
  //Customize the chart border and opacity
  border: { color: "#FF0000", width: 2 },
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Chart margin

The 3D chart's margin can be set from its container using the [margin](#) property.

INDEX.JS

```

var chartData = [
    { country: "USA", gold: 50, silver: 70, bronze: 45 },
    { country: "China", gold: 40, silver: 60, bronze: 55 },
    { country: "Japan", gold: 70, silver: 60, bronze: 50 },
    { country: "Australia", gold: 60, silver: 56, bronze: 40 },
    { country: "France", gold: 50, silver: 45, bronze: 35 },
    { country: "Germany", gold: 40, silver: 30, bronze: 22 },
    { country: "Italy", gold: 40, silver: 35, bronze: 37 },
    { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart3D = new ej.charts.Chart3D({
    primaryXAxis: {
        valueType: 'Category',
    },
    series:[{
        dataSource: chartData,
        xName: 'country', yName: 'gold', type: 'Column',
    }],
    title: 'Olympic Medals',
    wallColor: 'transparent',
    enableRotation: true,
    rotation: 7,
    tilt: 10,
    depth: 100,
    background: 'skyblue',
    border: { color: "#FF0000", width: 2 },
    //Change chart margin to left, right, top and bottom
    margin: { left: 40, right: 40, top: 40, bottom: 40 },
}, '#element');

```

INDEX.HTML


```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Animation

To customize the animation for a particular series, the [animation](#) property can be used. It can be enabled or disabled by using the [enable](#) property. The [duration](#) property specifies the duration of an animation and the [delay](#) property allows us to start the animation at desire time.

INDEX.JS

```

var chartData = [
  { country: "USA", gold: 50, silver: 70, bronze: 45 },
  { country: "China", gold: 40, silver: 60, bronze: 55 },
  { country: "Japan", gold: 70, silver: 60, bronze: 50 },
  { country: "Australia", gold: 60, silver: 56, bronze: 40 },
  { country: "France", gold: 50, silver: 45, bronze: 35 },
  { country: "Germany", gold: 40, silver: 30, bronze: 22 },
  { country: "Italy", gold: 40, silver: 35, bronze: 37 },
  { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart3D = new ej.charts.Chart3D({
  primaryXAxis: {
    valueType: 'Category',
  },
  series:[{
    dataSource: chartData,
    xName: 'country', yName: 'gold', type: 'Column',
    //Animation for chart series
    animation:{
      enable: true,
      duration: 2000,

```

```

        delay: 200
    }
    }],
    title: 'Olympic Medals',
    wallColor: 'transparent',
    enableRotation: true,
    rotation: 7,
    tilt: 10,
    depth: 100,
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Chart rotation

The 3D chart can be rotated by using the [enableRotation](#) property.

INDEX.JS

```

var chartData = [
  { country: "USA", gold: 50, silver: 70, bronze: 45 },
  { country: "China", gold: 40, silver: 60, bronze: 55 },
  { country: "Japan", gold: 70, silver: 60, bronze: 50 },
  { country: "Australia", gold: 60, silver: 56, bronze: 40 },
  { country: "France", gold: 50, silver: 45, bronze: 35 },
  { country: "Germany", gold: 40, silver: 30, bronze: 22 },
  { country: "Italy", gold: 40, silver: 35, bronze: 37 },
  { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];

```

```

var chart3D = new ej.charts.Chart3D({
  primaryXAxis: {
    valueType: 'Category',
  },
  series:[{
    dataSource: chartData,
    xName: 'country', yName: 'gold', type: 'Column'
  }],
  title: 'Olympic Medals',
  wallColor: 'transparent',
  enableRotation: true,
  rotation: 7,
  tilt: 10,
  depth: 100,
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Title

The 3D chart can be given a title by using [title](#) property, to show the information about the data plotted.

INDEX.JS

```

var chartData = [
  { country: "USA", gold: 50, silver: 70, bronze: 45 },
  { country: "China", gold: 40, silver: 60, bronze: 55 },
  { country: "Japan", gold: 70, silver: 60, bronze: 50 },
  { country: "Australia", gold: 60, silver: 56, bronze: 40 },

```

```

    { country: "France", gold: 50, silver: 45, bronze: 35 },
    { country: "Germany", gold: 40, silver: 30, bronze: 22 },
    { country: "Italy", gold: 40, silver: 35, bronze: 37 },
    { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
  ];
  var chart3D = new ej.charts.Chart3D({
    primaryXAxis: {
      valueType: 'Category',
    },
    series:[{
      dataSource: chartData,
      xName: 'country', yName: 'gold', type: 'Column'
    }],
    title: 'Olympic Medals',
    wallColor: 'transparent',
    enableRotation: true,
    rotation: 7,
    tilt: 10,
    depth: 100,
  }, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
  var ele = document.getElementById('container');
  if(ele) {
    ele.style.visibility = "visible";
  }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Title position

By using the [position](#) property in [titleStyle](#), the [title](#) can be positioned at left, right, top or bottom of the 3D chart. The title is positioned at the top of the 3D chart, by default.

INDEX.JS

```

var chartData = [
  { country: "USA", gold: 50, silver: 70, bronze: 45 },
  { country: "China", gold: 40, silver: 60, bronze: 55 },
  { country: "Japan", gold: 70, silver: 60, bronze: 50 },
  { country: "Australia", gold: 60, silver: 56, bronze: 40 },
  { country: "France", gold: 50, silver: 45, bronze: 35 },
  { country: "Germany", gold: 40, silver: 30, bronze: 22 },
  { country: "Italy", gold: 40, silver: 35, bronze: 37 },
  { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart3D = new ej.charts.Chart3D({
  primaryXAxis: {
    valueType: 'Category',
  },
  series:[{
    dataSource: chartData,
    xName: 'country', yName: 'gold', type: 'Column'
  }],
  title: 'Olympic Medals',
  titleStyle: { position: 'Bottom' },
  wallColor: 'transparent',
  enableRotation: true,
  rotation: 7,
  tilt: 10,
  depth: 100,
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>

```

```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

The custom option is used to position the title anywhere in the 3D chart using [x](#) and [y](#) coordinates.

INDEX.JS

```
var chartData = [
  { country: "USA", gold: 50, silver: 70, bronze: 45 },
  { country: "China", gold: 40, silver: 60, bronze: 55 },
  { country: "Japan", gold: 70, silver: 60, bronze: 50 },
  { country: "Australia", gold: 60, silver: 56, bronze: 40 },
  { country: "France", gold: 50, silver: 45, bronze: 35 },
  { country: "Germany", gold: 40, silver: 30, bronze: 22 },
  { country: "Italy", gold: 40, silver: 35, bronze: 37 },
  { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart3D = new ej.charts.Chart3D({
  primaryXAxis: {
    valueType: 'Category',
  },
  series:[{
    dataSource: chartData,
    xName: 'country', yName: 'gold', type: 'Column'
  }],
  title: 'Olympic Medals',
  titleStyle: {
    position: 'Custom',
    x: 300,
    y: 60
  },
  wallColor: 'transparent',
  enableRotation: true,
  rotation: 7,
  tilt: 10,
  depth: 100,
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>
```

```

<div id="container">
  <div id="element"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Title alignment

The title can be aligned to the near, far, or center of the 3D chart by using the [textAlignment](#) property.

INDEX.JS

```

var chartData = [
  { country: "USA", gold: 50, silver: 70, bronze: 45 },
  { country: "China", gold: 40, silver: 60, bronze: 55 },
  { country: "Japan", gold: 70, silver: 60, bronze: 50 },
  { country: "Australia", gold: 60, silver: 56, bronze: 40 },
  { country: "France", gold: 50, silver: 45, bronze: 35 },
  { country: "Germany", gold: 40, silver: 30, bronze: 22 },
  { country: "Italy", gold: 40, silver: 35, bronze: 37 },
  { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart3D = new ej.charts.Chart3D({
  primaryXAxis: {
    valueType: 'Category',
  },
  series:[{
    dataSource: chartData,
    xName: 'country', yName: 'gold', type: 'Column'
  }],
  title: 'Olympic Medals',
  titleStyle: {
    textAlignment: 'Far'
  },
  wallColor: 'transparent',
  enableRotation: true,
  rotation: 7,
  tilt: 10,
  depth: 100,
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Title customization

The [titleStyle](#) property of the 3D chart provides options to customize the title by using the following properties.

- [size](#) - Specifies the size of the title.
- [color](#) - Specifies the color for the title.
- [fontFamily](#) - Specifies the font family for the title.
- [fontWeight](#) - Specifies the font weight of the title.
- [fontStyle](#) - Specifies the font style for the title.
- [opacity](#) - Specifies the opacity for the color of the title.
- [textAlignment](#) - Specifies the alignment of the title.
- [textOverflow](#) - Specifies the overflow of the title.

INDEX.JS

```

var chartData = [
    { country: "USA", gold: 50, silver: 70, bronze: 45 },
    { country: "China", gold: 40, silver: 60, bronze: 55 },
    { country: "Japan", gold: 70, silver: 60, bronze: 50 },
    { country: "Australia", gold: 60, silver: 56, bronze: 40 },
    { country: "France", gold: 50, silver: 45, bronze: 35 },
    { country: "Germany", gold: 40, silver: 30, bronze: 22 },
    { country: "Italy", gold: 40, silver: 35, bronze: 37 },
    { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart3D = new ej.charts.Chart3D({
    primaryXAxis: {
        valueType: 'Category',
    },
    series:[{
        dataSource: chartData,
        xName: 'country', yName: 'gold', type: 'Column'
    }],
    title: 'Olympic Medals',

```



```

    titleStyle: {
      size: '18px', color: 'Red', textOverflow: 'Wrap'
    },
    wallColor: 'transparent',
    enableRotation: true,
    rotation: 7,
    tilt: 10,
    depth: 100,
  }, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Accessibility in EJ2 JavaScript 3D Chart control

Accessibility is achieved in the 3D chart control through WAI-ARIA standard and keyboard navigation. The 3D chart features can be effectively accessed through assistive technologies such as screen readers.

WAI-ARIA

WAI-ARIA (Accessibility Initiative – Accessible Rich Internet Applications) defines a way to increase the accessibility of web pages, dynamic content, and user interface components developed with AJAX, HTML, JavaScript, and related technologies. ARIA provides additional semantics to describe the role, state, and functionality of web components.

- img (role)
- button (role)
- region (role)

- aria-label (attribute)
- aria-hidden (attribute)
- aria-pressed (attribute)

Keyboard navigation

All the 3D chart actions can be controlled via keyboard keys. The applicable key combinations and their relative functionalities are listed below.

Interaction Keys | Description

Tab | Moves the focus to the next element in the chart.

Shift + Tab | Moves the focus to the previous element in the chart.

DownArrow | Moves the focus to the data point left side from the selected point.

UpArrow | Moves the focus to the data point right side from the selected point.

Left Arrow | Moves the focus to the next series in the chart.

Right Arrow | Moves the focus to the previous series in the chart.

ESC | Cancel the tooltip for the data point.

Enter/Space | Selects the data point in the series.

Down/Left Arrow | Moves the focus to the legend left side from the selected legend.

Up/Right Arrow | Moves the focus to the legend right side from the selected legend.

Enter/Space | Toggles the visibility of the corresponding series.

Ctrl + P | Print the chart.

Accordion

Expand mode in ##Platform_Name## Accordion control

- Single
- Multiple

Single

The property enables to expand only one Accordion item at a time. If you expand any new item, the previously expanded one is collapsed and new item changed to expanded state.

You can also choose which accordion pane is expanded state at initial rendering by enabling the [expanded](#) property on accordion items.

INDEX.JS

```
ej.base.enableRipple(true);
//Initialize Accordion component
var accordion = new ej.navigations.Accordion({
  expandMode: 'Single',
  items: [
```

```

    { header: 'ASP.NET', expanded: true, content: 'Microsoft ASP.NET is a
    set of technologies in the Microsoft .NET Framework for building Web
    applications and XML Web services.' },
    { header: 'ASP.NET MVC', content: 'The Model-View-Controller (MVC)
    architectural pattern separates an application into three main components:
    the model, the view, and the controller.' },
    { header: 'JavaScript', content: 'JavaScript (JS) is an interpreted
    computer programming language. It was originally implemented as part of web
    browsers so that client-side scripts could interact with the user, control
    the browser, communicate asynchronously, and alter the document content that
    was displayed.' },
  ]
});
//Render initialized Accordion component
accordion.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Accordion</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Toolbar Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
    <br><br>
    <div id="result"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Multiple

Default [expandMode](#) of the Accordion is **Multiple**. It enables you to expand more than one Accordion item at a time. Expand/collapse action can also be toggled by clicking on it again. For example, expanded item is collapsed when you click on it again.

INDEX.JS

```

ej.base.enableRipple(true);
//Initialize Accordion component
var accordion = new ej.navigations.Accordion({
  expandMode: 'Multiple',
  items: [
    { header: 'ASP.NET', content: 'Microsoft ASP.NET is a set of
technologies in the Microsoft .NET Framework for building Web applications
and XML Web services.' },
    { header: 'ASP.NET MVC', content: 'The Model-View-Controller (MVC)
architectural pattern separates an application into three main components:
the model, the view, and the controller.' },
    { header: 'JavaScript', content: 'JavaScript (JS) is an interpreted
computer programming language. It was originally implemented as part of web
browsers so that client-side scripts could interact with the user, control
the browser, communicate asynchronously, and alter the document content that
was displayed.' },
  ]
});
//Render initialized Accordion component
accordion.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Accordion</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Toolbar Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
    <br><br>
    <div id="result"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [How to keep single pane open always](#)

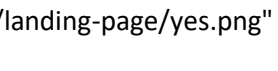
Accessibility in ##Platform_Name## Accordion control

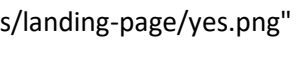
The Accordion control has been designed keeping in mind the [WAI-ARIA](#) specifications, by applying the prompt WAI-ARIA roles, states, and properties along with the keyboard support. Thus, making it usable for people who use assistive WAI-ARIA Accessibility supports that is achieved through the attributes like `aria-labelledby`. It helps to provides information about the elements in a document for assistive technology. The control implements the keyboard navigation support by following the [WAI-ARIA practices](#) and tested in major screen readers.

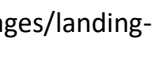
The accessibility compliance for the Accordion control is outlined below.

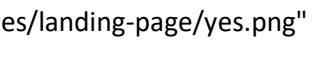
| Accessibility Criteria | Compatibility |

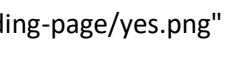
| -- | -- |

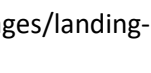
| [WCAG 2.2](#) Support |  |

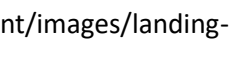
| [Section 508](#) Support |  |

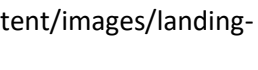
| Screen Reader Support |  |

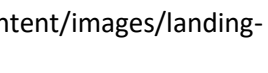
| Right-To-Left Support |  |

| Color Contrast |  |

| Mobile Device Support |  |

| Keyboard Navigation Support |  |

| [Accessibility Checker](#) Validation |  |

| [Axe-core](#) Accessibility Validation |  |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<div> - All features of the control meet the requirement.</div>

<div> - Some features of the control do not meet the requirement.</div>

<div> - The control does not meet the requirement.</div>

ARIA attributes

| Roles and Attributes | Functionalities

| role | **Button:** Attribute is set to the Accordion header elements to indicate that the element can be used to toggle the visibility of the associated content section, describing the actual role of the element.
 Region: Attribute is set to the Accordion panel elements to create a landmark region that contains the currently expanded accordion panel, describing the actual role of the element.
|

| aria-labelledby | Attribute is set to content (panel) and it points to the corresponding Accordion header. |

| aria-controls | Attribute is set to the header and it points to the corresponding Accordion content. |

| aria-expanded | Attribute is set to the Accordion header elements to indicates the expand state of the Accordion Item. Default value of this attribute is **false**. If an item is expanded, the attribute value changes to 'true'. |

| aria-hidden | Attribute is set to the Accordion panel elements to indicates the content visible state of the Accordion Item. Default value of this attribute is **true**. If an item content is visible, the attribute value changes to **false**. |

| aria-disabled | It indicates the disabled state of the Accordion and its items. |

Keyboard interaction

Keyboard navigation is enabled by default. Possible keys are:

Key	Description	
Space or Enter	When focus is on the Accordion header, click on the focused element makes the element to expand and collapse.	
Down Arrow	Focus the next Accordion header.	
Up Arrow	Focus the previous Accordion header.	
Home	Focus the first Accordion header.	
End	Focus the last Accordion header.	
Tab	To Move focus through the interactive elements.	

| Shift + Tab | To Move focus through the interactive elements. |

Ensuring accessibility

The Accordion control accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Accordion control is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Accordion control with accessibility tools.

See also

- [Accessibility in Syncfusion components](#)

Style in ##Platform_Name## Accordion control

The following content provides the exact CSS structure that can be used to modify the control's appearance based on user preference.

Customizing Accordion

Use the following CSS to customize the Accordion.

```
`css
.e-accordion {
border: 5px solid rgb(173, 255, 47);
}
`
```

Customizing the list items

Use the following CSS to customize the items of Accordion.

```
`css
.e-accordion .e-acrdn-item {
text-align: center;
color: pink;
background-color: #2fa1ff;
}
`
```

Customizing Accordion's header

Use the following CSS to customize the header of Accordion control.

```
`css
.e-accordion .e-acrdn-item.e-select > .e-acrdn-header {
background: #2fa1ff !important;
justify-content: center;
}
`
```

,

Customizing Accordion's expand and collapse icons

Use the following CSS to customize the expand and collapse icons of Accordion control.

```
`css
.e-accordion .e-acrdn-item .e-acrdn-header .e-toggle-icon .e-icons {
color: pink;
}
,
```

Customizing the hover state of Accordion control

Use the following CSS to customize the accordion item when hovering.

```
`css
.e-accordion .e-acrdn-item .e-acrdn-header:hover {
border: 2px solid gray;
}
,
```

Customizing selected item of Accordion control

Use the following CSS to customize the selected accordion item.

```
`css
.e-accordion .e-acrdn-item.e-select.e-active>.e-acrdn-header,
.e-accordion .e-acrdn-item.e-select.e-item-focus>.e-acrdn-header {
background-color: rgb(0, 15, 100) !important;
}
,
```

Use the following CSS to customize the selected accordion item text.

```
`css
.e-accordion .e-acrdn-item.e-select.e-active>.e-acrdn-header .e-acrdn-header-content,
.e-accordion .e-acrdn-item.e-select.e-item-focus>.e-acrdn-header .e-acrdn-header-content {
color: #2fa1ff !important;
}
,
```

How To

Set the nested accordion in ##Platform_Name## Accordion control

Accordion supports to render **nested** level of Accordion by using content property. You can give nested Accordion content inside the parent Accordion content property by using **id** of nested element. The nested Accordion can be rendered with the use of provided events, such as [clicked](#) and [expanding](#).

INDEX.JS

```

ej.base.enableRipple(true);
//Initialize Accordion component
var nestAcrdn_vid;
var nestAcrdn_mus;
var nestAcrdn_musNew;
var nestAcrdn_img;
var accordion = new ej.navigations.Accordion({
    expanding: expanding,
    items: [
        { header: 'Video', content: '<div id="nested_video"></div>' },
        { header: 'Music', content: '<div id="nested_music"></div>' },
        { header: 'Images', content: '<div id="nested_images"></div>' },
    ]
});
//Render initialized Accordion component
accordion.appendTo('#element');
function clicked(e) {
    var ele = e.originalEvent.target;
    if (ele.querySelectorAll('.e-accordion').length > 0) {
        return;
    }
    if (!document.getElementById("nested_musicNew").classList.contains("e-accordion")) {
        nestAcrdn_musNew = new ej.navigations.Accordion({
            items: [
                { header: 'New Track1' },
                { header: 'New Track2' }
            ]
        }, '#nested_musicNew');
    }
}
function expanding(e) {
    if (e.isExpanded && [].indexOf.call(this.items, e.item) === 0) {
        if (e.element.querySelectorAll('.e-accordion').length > 0) {
            return;
        }
        nestAcrdn_vid = new ej.navigations.Accordion({
            items: [
                { header: 'Video Track1' },
                { header: 'Video Track2' }
            ]
        }, '#nested_video');
    }
    if (e.isExpanded && [].indexOf.call(this.items, e.item) === 1) {
        if (e.element.querySelectorAll('.e-accordion').length > 0) {
            return;
        }
        nestAcrdn_mus = new ej.navigations.Accordion({
            clicked: clicked,
            items: [
                { header: 'Music Track1' },
                { header: 'Music Track2' },
                { header: 'Music New', content: '<div id="nested_musicNew"></div>' }
            ]
        }, '#nested_music');
    }
}

```

```

    }
    if (e.isExpanded && [].indexOf.call(this.items, e.item) === 2) {
        if (e.element.querySelectorAll('.e-accordion').length > 0) {
            return;
        }
        nestAcrdn_img = new ej.navigations.Accordion({
            items: [
                { header: 'Track1' },
                { header: 'Track2' },
            ]
        }, '#nested_images');
    }
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Accordion</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Toolbar Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
    <br><br>
    <div id="result"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Load content through post in ##Platform_Name## Accordion control

Accordion supports to load external contents through **AJAX** library. Refer the below steps.

- Import the **Ajax** module from **ej2-base** and initialize with URL path.
- Get data from the Ajax Success event to initialize Accordion with retrieved external path data.

INDEX.JS

```

ej.base.enableRipple(true);
var ajax = new ej.base.Ajax('./ajax.html', 'GET', true);
ajax.send().then();
ajax.onSuccess = function (data) {
    var ctn2 = data;
    //Initialize Accordion component
    var accordion = new ej.navigations.Accordion({
        items: [
            { header: 'Department', content: '#acrdnContent1' },
            { header: 'Platform', content: '#acrdnContent2' },
            { header: 'Employee Details', content: ctn2 }
        ]
    });
    //Render initialized Accordion component
    accordion.appendTo('#element');
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Accordion</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Toolbar Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
        <br><br>
        <div id="result"></div>
        <div id="acrdnContent1" style="display:none">
            <ul style="margin : 0px;padding:0px 16px; list-style-type:
none">
                <li>Testing</li>
                <li>Development</li>
            </ul>
        </div>
        <div id="acrdnContent2" style="display:none">
            <ul style="margin : 0px;padding:0px 16px; list-style-type:
none">
                <li>Mobile</li>
                <li>Web</li>
            </ul>

```

```

        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Set custom animation in ##Platform_Name## Accordion control

Accordion supports custom animations for both expand and collapse actions from the provided animation option of [Animation](#) library. The [animation](#) property also allows you to set [easing](#), [duration](#), and various other effects of your choice.

Default animation is given as [SlideDown](#) for expanding the panel using [expand](#) animation property and [SlideUp](#) for collapsing the panel using [collapse](#) animation property. You can also disable the animation by setting animation [effect](#) as [none](#).

The sample demonstrates some types of animation that suits for Accordion. You can check all the animation effects [here](#).

INDEX.JS

```

ej.base.enableRipple(true);
//Initialize Accordion component
var accordion = new ej.navigations.Accordion({
    items: [
        { header: 'ASP.NET', expanded: 'true', content: 'Microsoft ASP.NET
is a set of technologies in the Microsoft .NET Framework for building Web
applications and XML Web services.' },
        { header: 'ASP.NET MVC', content: 'The Model-View-Controller (MVC)
architectural pattern separates an application into three main components:
the model, the view, and the controller.' },
        { header: 'JavaScript', content: 'JavaScript (JS) is an interpreted
computer programming language. It was originally implemented as part of web
browsers so that client-side scripts could interact with the user, control
the browser, communicate asynchronously, and alter the document content that
was displayed.' },
    ]
});
//Render initialized Accordion component
accordion.appendTo('#element');
var listObjExpand = new ej.dropdowns.DropDownList({
    index: 0,
    placeholder: 'Select a animate type',
    popupHeight: '150px',
    change: () => { valueChange(); }
});
listObjExpand.appendTo('#expandAnimation');
valueChange();

var listObjCollapse = new ej.dropdowns.DropDownList({
    index: 1,

```

```

        placeholder: 'Select a animate type',
        popupHeight: '150px',
        change: () => { valueChange1(); }
    });
    listObjCollapse.appendTo('#collapseAnimation');
    valueChange1();
    function valueChange() {
        accordion.animation.expand.effect = (listObjExpand.value);
    }
    function valueChange1() {
        accordion.animation.collapse.effect = (listObjCollapse.value);
    }

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Accordion</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Toolbar Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
>

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="default" style="padding-bottom:75px;">
            <div class="row">
                <div class="col-xs-6 col-sm-6 col-lg-6 col-md-6">
                    <label> Expand Animation </label>
                </div>
                <div class="col-xs-6 col-sm-6 col-lg-6 col-md-6">
                    <select id="expandAnimation">
                        <option value="SlideDown">SlideDown</option>
                        <option value="SlideUp">SlideUp</option>
                        <option value="FadeIn">FadeIn</option>
                        <option value="FadeOut">FadeOut</option>
                        <option value="FadeZoomIn">FadeZoomIn</option>
                        <option value="FadeZoomOut">FadeZoomOut</option>
                        <option value="ZoomIn">ZoomIn</option>
                        <option value="ZoomOut">ZoomOut</option>
                        <option value="None">None</option>
                    </select>
                </div>
            </div>
        </div>
    </div>

```

```

        <div class="row">
            <div class="col-xs-6 col-sm-6 col-lg-6 col-md-6">
                <label> Collapse Animation </label>
            </div>
            <div class="col-xs-6 col-sm-6 col-lg-6 col-md-6">
                <select id="collapseAnimation">
                    <option value="SlideDown">SlideDown</option>
                    <option value="SlideUp">SlideUp</option>
                    <option value="FadeIn">FadeIn</option>
                    <option value="FadeOut">FadeOut</option>
                    <option value="FadeZoomIn">FadeZoomIn</option>
                    <option value="FadeZoomOut">FadeZoomOut</option>
                    <option value="ZoomIn">ZoomIn</option>
                    <option value="ZoomOut">ZoomOut</option>
                    <option value="None">None</option>
                </select>
            </div>
        </div>
        <div id="element"></div>
        <br><br>
        <div id="result"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

To keep single pane open always in ##Platform_Name## Accordion control

By default, all Accordion panels are collapsible. You can customize the Accordion to keep one panel as expanded state always. This is applicable for **Single** expand mode.

INDEX.JS

```

ej.base.enableRipple(true);
//Initialize Accordion component
var clickEle;
var accordion = new ej.navigations.Accordion({
    expandMode: 'Single',
    clicked: clicked,
    expanding: beforeExpand,
    items: [
        { header: 'ASP.NET', expanded: 'true', content: 'Microsoft ASP.NET is a set of technologies in the Microsoft .NET Framework for building Web applications and XML Web services.' },
        { header: 'ASP.NET MVC', content: 'The Model-View-Controller (MVC) architectural pattern separates an application into three main components: the model, the view, and the controller.' },
        { header: 'JavaScript', content: 'JavaScript (JS) is an interpreted computer programming language. It was originally implemented as part of web browsers so that client-side scripts could interact with the user, control

```

```

the browser, communicate asynchronously, and alter the document content that
was displayed.' },
]
});
//Render initialized Accordion component
accordion.appendTo('#element');
function clicked (e) {
    clickEle = e.originalEvent.target.closest('.e-acrdn-header');
}
function beforeExpand (e) {
    var expandCount = this.element.querySelectorAll('.e-selected').length;
    var ele = this.element.querySelectorAll('.e-selected')[0];
    if (ele) {
        ele = ele.firstChild
    }
    if (expandCount === 1 && ele === clickEle) {
        e.cancel = true;
    }
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Accordion</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Toolbar Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
        <br><br>
        <div id="result"></div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

Create wizard using accordion in `##Platform_Name##` Accordion control

Accordion items can be disabled dynamically by passing the index and boolean value with the [enableItem](#) method and also dynamically expand the item using [expandItem](#) method.

The below Wizard sample is designed for Online Shopping model. In this, each Accordion item is integrated with required components to fill

the details and designed for getting user details and making payment at the end. Each field is provided with validation for all mandatory

option to enable/disable to next Accordion. In below sample, accordion items can be disabled dynamically with [enableItem](#) method using [created](#) event.

INDEX.JS

```
ej.base.enableRipple(true);
var success = 'Your payment successfully processed';
var email_alert = 'Enter valid email address';
var mobile_alert = 'Mobile number length should be 10';
var card_alert = 'Card number length should be 16';
var cvv_alert = 'CVV number length should be 3';
var mobile = new ej.inputs.NumericTextBox({
  placeholder: 'Mobile',
  floatLabelType: 'Auto',
  format: '0',
  showSpinButton: false
});
mobile.appendTo('#mobile');
var cardNum = new ej.inputs.NumericTextBox({
  placeholder: 'Card No',
  floatLabelType: 'Auto',
  format: '0',
  showSpinButton: false
});
cardNum.appendTo('#cardNo');
var datepicker = new ej.calendars.DatePicker({
  width: '100%',
  format: 'MM/yyyy',
  floatLabelType: 'Auto',
  placeholder: 'Expiry Date'
});
datepicker.appendTo('#expiry');
var cvv = new ej.inputs.NumericTextBox({
  placeholder: 'CVV',
  floatLabelType: 'Auto',
  format: '0',
  showSpinButton: false
});
cvv.appendTo('#CVV');
var alertDialog = new ej.popups.Dialog({
  header: 'Alert',
  width: 200,
  isModal: true,
  content: '',
  target: document.body,
  buttons: [{
    buttonModel: { content: 'Ok', isPrimary: true },
```



```

        click: (() => {
            alertDialog.hide();
            if (acrdnObj.expandedIndices[0] === 2 && alertDialog.content ===
success) {
                acrdnObj.enableItem(0, true);
                acrdnObj.enableItem(1, false);
                acrdnObj.enableItem(2, false);
                acrdnObj.expandItem(true, 0);
            }
        })
    }
});
alertDialog.appendTo('#alertDialog');
alertDialog.hide();
var acrdnObj = new ej.navigations.Accordion({
    expanding: expand,
    created: () => {
    },
    items: [
        { header: 'Sign In', content: '#Sign_In_Form', expanded: true },
        { header: 'Delivery Address', content: '#Address_Fill' },
        { header: 'Card Details', content: '#Card_Fill' },
    ]
});
acrdnObj.appendTo('#element');
function checkMail(mail) {
    if (/^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,3})+$/ .test(mail)) {
        return true;
    } else {
        alertDialog.content = email_alert;
        alertDialog.show();
        return false;
    }
}
function checkMobile(mobile) {
    if (mobile.match(/^d{10}$/)) {
        return true;
    } else {
        alertDialog.content = mobile_alert;
        alertDialog.show();
        return false;
    }
}
function checkCardNo(cardNo) {
    if (cardNo.match(/^d{16}$/)) {
        return true;
    } else {
        alertDialog.content = card_alert;
        alertDialog.show();
        return false;
    }
}
function checkCVV(cvv) {
    if (cvv.match(/^d{3}$/)) {
        return true;
    } else {
        alertDialog.content = cvv_alert;
    }
}

```

```

        alertDialog.show();
        return (false);
    }
}

function expand(e) {
    if (e.name === 'expanding' && [].indexOf.call(this.items, e.item) === 0) {
        document.getElementById('Continue_Btn').onclick = (e) => {
            var email = document.getElementById('email');
            var password = document.getElementById('password');
            if (email.value !== '' && password.value !== '') {
                if (checkMail(email.value)) {
                    email.value = password.value = '';
                    acrdnObj.enableItem(1, true);
                    acrdnObj.enableItem(0, false);
                    acrdnObj.expandItem(true, 1);
                }
                document.getElementById('err1').classList.remove('show');
            } else {
                document.getElementById('err1').classList.add('show');
            }
        };
    } else if (e.name === 'expanding' && [].indexOf.call(this.items, e.item) === 1) {
        document.getElementById('Continue_BtnAdr').onclick = (e) => {
            var name = document.getElementById('name');
            var address = document.getElementById('address');
            var mobile = document.getElementById('mobile');
            if ((name.value !== '') && (address.value !== '') && (mobile.value !== '')) {
                if (checkMobile(mobile.value)) {
                    acrdnObj.enableItem(2, true);
                    acrdnObj.enableItem(1, false);
                    acrdnObj.expandItem(true, 2);
                }
                document.getElementById('err2').classList.remove('show');
            } else {
                document.getElementById('err2').classList.add('show');
            }
        };
    } else if (e.name === 'expanding' && [].indexOf.call(this.items, e.item) === 2) {
        document.getElementById('Back_Btn').onclick = (e) => {
            acrdnObj.enableItem(1, true);
            acrdnObj.enableItem(2, false);
            acrdnObj.expandItem(true, 1);
        };
        document.getElementById('Save_Btn').onclick = (e) => {
            var cardHolder = document.getElementById('cardHolder');
            var expiry = document.getElementById('expiry');
            var cardNo = document.getElementById('cardNo');
            var cvv = document.getElementById('CVV');
            if ((cardNo.value !== '') && (cardHolder.value !== '') && (expiry.value !== '') && (cvv.value !== '')) {
                if (checkCardNo(cardNo.value)) {
                    if (checkCVV(cvv.value)) {
                        alertDialog.content = success;
                        alertDialog.show();
                    }
                }
            }
        };
    }
}

```

```

    }
  }
  document.getElementById('err3').classList.remove('show');
} else {
  document.getElementById('err3').classList.add('show');
}
};
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Accordion</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Toolbar Controls">
  <meta name="author" content="Syncfusion">
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
  <link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="template_title"> Online Shopping Payment Module</div>
    <div id="element"></div>
    <div id="result"></div>
  </div>
  <div id="Sign_In_Form" style="display: none; padding: 3px 0">
    <form id="formId">
      <div class="form-group">
        <div class="e-float-input">
          <input type="text" id="email" name="Email" required="">
          <span class="e-float-line"></span>
          <label class="e-float-text" for="email">Email</label>
        </div>
        <div class="e-float-input">
          <input class="e-input" id="password" type="password"
name="Password" required="">
          <span class="e-float-line"></span>
          <label class="e-float-text"
for="password">Password</label>
        </div>
      </div>
    </form>
    <div style="text-align: center">

```

```

        <button class="e-btn" id="Continue_Btn">Continue</button>
        <div id="err1">* Please fill all fields</div>
    </div>
</div>
<div id="Address_Fill" style="display: none; padding: 3px 0">
    <form id="formId_Address">
        <div class="form-group">
            <div class="e-float-input">
                <input type="text" id="name" name="Name" required="">
                <span class="e-float-line"></span>
                <label class="e-float-text" for="name">Name</label>
            </div>
        </div>
        <div class="form-group">
            <div class="e-float-input">
                <input type="text" id="address" name="Address"
required="">
                <span class="e-float-line"></span>
                <label class="e-float-text"
for="address">Address</label>
            </div>
        </div>
        <div class="form-group">
            <input type="text" id="mobile">
        </div>
    </form>
    <div style="text-align: center">
        <button class="e-btn" id="Continue_BtnAdr">Continue</button>
        <div id="err2">* Please fill all fields</div>
    </div>
</div>
<div id="Card_Fill" style="display: none; padding: 3px 0;">
    <div class="col-xs-6 col-sm-6 col-lg-6 col-md-6">
        <div class="form-group">
            <input type="text" id="cardNo">
        </div>
    </div>
    <div class="col-xs-6 col-sm-6 col-lg-6 col-md-6">
        <div class="form-group">
            <div class="e-float-input">
                <input type="text" id="cardHolder" name="cardHolder"
required="">
                <span class="e-float-line"></span>
                <label class="e-float-text" for="cardHolder">CardHolder
Name</label>
            </div>
        </div>
    </div>
    <div class="col-xs-6 col-sm-6 col-lg-6 col-md-6">
        <input id="expiry">
    </div>
    <div class="col-xs-6 col-sm-6 col-lg-6 col-md-6">
        <div class="form-group">
            <input type="text" id="CVV">
        </div>
    </div>
    <div style="text-align: center">

```

```

        <button class="e-btn" id="Back_Btn">Back</button>
        <button class="e-btn" id="Save_Btn">Save</button>
        <div id="err3">* Please fill all fields</div>
    </div>
</div>
<div id="alertDialog"></div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Load accordion with data source in ##Platform_Name## Accordion control

You can bind any data object to Accordion items, by mapping it to [header](#) and [content](#) property.

In the below demo, Data is fetched from an OData service using DataManager. The result data is formatted as a JSON object with [header](#) and [content](#) fields, which is set to items property of Accordion.

INDEX.JS

```

var itemsData = [];
var mapping = { header: 'FirstName', content: 'Notes' };
const SERVICE_URI =
'https://services.odata.org/V4/Northwind/Northwind.svc/Employees';
new ej.data.DataManager({ url: SERVICE_URI, adaptor: new
ej.data.ODataV4Adaptor, crossDomain: true })
    .executeQuery(new ej.data.Query().range(1, 4)).then(function (e) {
        var result = e.result;
        for(var i = 0; i < result.length; i++) {
            itemsData.push({ header: result[i][mapping.header], content:
result[i][mapping.content] });
        }
        //Initialize Accordion component
        var accordion = new ej.navigations.Accordion({
            items: itemsData
        });
        //Render initialized Accordion component
        accordion.appendTo('#element');
    });

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Accordion</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Toolbar Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">

```

```
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
        <br><br>
        <div id="result"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Load accordion items dynamically in ##Platform_Name## Accordion control

Accordion items can be added dynamically by passing the item and index value with the [addItem](#) method.

In the following demo, new items are added dynamically when you expand an Accordion header using [expanded](#) event.

- Data is fetched from the data source and it is formatted as a JSON object with **header** and **content** fields.
- Here last index is calculated to append the new Accordion item at the end and the number of items are limited to 10, since the data bound have only 10 items.

INDEX.JS

```
ej.base.enableRipple(true);
//Initialize Accordion component
var dbFlag = 0;
var dynamciAcrdnCount = 2;
var accordionData = [
    {
        header: ' ASP.NET Razor ',
        content:
            ' Razor is an ASP.NET programming syntax used to create dynamic web
pages with the C# or Visual Basic .NET programming languages. Razor was in
development in June 2010 and was released for Microsoft Visual Studio 2010
in January 2011. Razor is a simple-syntax view engine and was released as
part of MVC 3 and the WebMatrix tool set. '
    },
    {
        header: ' EmberJS ',
        content:
```

```
' EmberJS is an open-source JavaScript web framework, based on the
Model-view-viewmodel (MVVM) pattern. It allows developers to create scalable
single-page web applications by incorporating common idioms and best
practices into the framework. '
},
{
  header: ' Hypertext Markup Language ',
  content:
    ' Hypertext Markup Language (HTML) is the standard markup language for
creating web pages and web applications. With Cascading Style Sheets (CSS)
and JavaScript, it forms a triad of cornerstone technologies for the World
Wide Web. '
},
{
  header: ' HTML5 ',
  content:
    ' HTML5 is a markup language used for structuring and presenting
content on the World Wide Web. It is the fifth and current major version of
the HTML standard. '
},
{
  header: ' JavaScript ',
  content:
    ' JavaScript (JS) is an interpreted computer programming language. '
},
{
  header: ' JSP ',
  content:
    ' JavaServer Pages (JSP) is a technology that helps software
developers create dynamically generated web pages based on HTML, XML, or
other document types. Released in 1999 by Sun Microsystems, JSP is similar
to PHP and ASP, but it uses the Java programming language. '
},
{
  header: ' Perl ',
  content:
    ' Perl is a family of high-level, general-purpose, interpreted,
dynamic programming languages. Perl was originally developed by Larry Wall
in 1987 as a general-purpose Unix scripting language to make report
processing easier. '
},
{
  header: ' PHP ',
  content:
    ' PHP is a server-side scripting language designed for web development
but also used as a general-purpose programming language. Originally created
by Rasmus Lerdorf in 1994, the PHP reference implementation is now produced
by The PHP Group. PHP originally stood for Personal Home Page, but it now
stands for the recursive acronym PHP: Hypertext Preprocessor '
},
{
  header: ' Ruby ',
  content:
    ' Ruby is a dynamic, reflective, object-oriented, general-purpose
programming language. It supports multiple programming paradigms, including
functional, object-oriented, and imperative. It also has a dynamic type
system and automatic memory management. '
```

```

    },
    {
        header: ' Typescript ',
        content:
            ' TypeScript is an open-source programming language developed and
            maintained by Microsoft. It is a strict syntactical superset of JavaScript,
            and adds optional static typing to the language. '
    }
];
var acrdnObj = new ej.navigations.Accordion({
    expanded: expanded,
    items: [
        {
            header: 'ASP.NET',
            content:
                'Microsoft ASP.NET is a set of technologies in the Microsoft .NET
                Framework for building Web applications and XML Web services.'
        },
        {
            header: ' ASP.NET Core ',
            content:
                ' ASP.NET Core is a free and open-source web framework, and the next
                generation of ASP.NET, developed by Microsoft and the community. It is a
                modular framework that runs on both the full .NET Framework, on Windows, and
                the cross-platform .NET Core.'
        },
        {
            header: 'ASP.NET MVC',
            content:
                'The Model-View-Controller (MVC) architectural pattern separates an
                application into three main components: the model, the view, and the
                controller.'
        }
    ]
});
//Render initialized Accordion component
acrdnObj.appendTo('#Accordion_default');
function expanded(e) {
    //Find the expanded state item in accordion
    var Elementindex = document.getElementsByClassName(
        'e-expand-state e-selected e-active'
    )[0];
    //Check the expand state item and currently selected item index. If it is
    expanded state add dyanmic items or else does nothing
    if (
        [].slice.call(e.element.parentElement.children).indexOf(e.element) ==
        [].slice.call(e.element.parentElement.children).indexOf(Elementindex)
    ) {
        var array = accordionData;
        for (var i = 0; i < dynamciAcrdnCount; i++) {
            if (dbFlag === array.length) {
                //checking whether all the items in data source added
                return;
            }
            // Item object and the index argument passed into the additem method
            to add a new accordion
            acrdnObj.addItem(array[dbFlag], acrdnObj.items.length);
        }
    }
}

```



```

        ++dbFlag;
    }
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Accordion</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Toolbar Controls">
  <meta name="author" content="Syncfusion">
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="control-section">
      <div class="control_wrapper accordion-control-section">
        <div id="Accordion_default"></div>
      </div>
    </div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Add icon to accordion header in ##Platform_Name## Accordion control

You can add the icon custom css class to the Accordion header using '[iconCss](#)' property and also add css styles to the defined class. The accordion icon element is rendered before the header text in the DOM element.

INDEX.JS

```

ej.base.enableRipple(true);
//Initialize Accordion component
var accordion = new ej.navigations.Accordion({
  items: [

```

```

        { header: 'Athletics', iconCss: 'e-athletics e-acrdn-icons',
content: '#athletics', expanded: true },
        { header: 'Water Games', iconCss: 'e-water-game e-acrdn-icons',
content: '#water_games' },
        { header: 'Racing', iconCss: 'e-racing-games e-acrdn-icons',
content: '#racing_games' },
        { header: 'Indoor Games', iconCss: 'e-indoor-games e-acrdn-
icons', content: '#indoor_games' }
    ]
});
//Render initialized Accordion component
accordion.appendTo('#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Accordion</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Toolbar Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
    <div id="athletics" style="display:none">
      <li><span class="e-acrdn-icons e-content-icon marathon"></span>
Marathon</li>
      <li><span class="e-acrdn-icons e-content-icon javelin"></span>
Javelin Throw</li>
      <li><span class="e-acrdn-icons e-content-icon discus"></span> Discus
Throw</li>
      <li><span class="e-acrdn-icons e-content-icon highjump"></span> High
Jump</li>
      <li><span class="e-acrdn-icons e-content-icon longjump"></span> Long
Jump</li>
    </div>
    <div id="water_games" style="display:none">
      <li><span class="e-acrdn-icons e-content-icon dive"></span>
Diving</li>
      <li><span class="e-acrdn-icons e-content-icon swimming"></span>
Swimming</li>
      <li><span class="e-acrdn-icons e-content-icon marathan_swim"></span>
Marathon Swimming</li>
      <li><span class="e-acrdn-icons e-content-icon sync_swim"></span>
Synchronized Swimming</li>
```

```

        <li><span class="e-acrdn-icons e-content-icon waterpolo"></span>
Water Polo</li>
    </div>
    <div id="racing_games" style="display:none">
        <li><span class="e-acrdn-icons e-content-icon cycle_BMX"></span>
Cycling BMX</li>
        <li><span class="e-acrdn-icons e-content-icon
cycle_Mountain"></span> Cycling Mountain Bike</li>
        <li><span class="e-acrdn-icons e-content-icon cycle"></span> Cycle
Racing</li>
        <li><span class="e-acrdn-icons e-content-icon sailing"></span>
Sailing</li>
        <li><span class="e-acrdn-icons e-content-icon rowing"></span>
Rowing</li>
    </div>
    <div id="indoor_games" style="display:none">
        <li><span class="e-acrdn-icons e-content-icon tennis"></span> Table
Tennis</li>
        <li><span class="e-acrdn-icons e-content-icon badminton"></span>
Badminton</li>
        <li><span class="e-acrdn-icons e-content-icon volleyball"></span>
Volleyball</li>
        <li><span class="e-acrdn-icons e-content-icon boxing"></span>
Boxing</li>
        <li><span class="e-acrdn-icons e-content-icon swimming_In"></span>
Swimming</li>
    </div>
    <br><br>
    <div id="result"></div>
</div>
<style>
@font-face {
    font-family: 'acrdn-icons';
    src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAIAIAAAwAgTlMvMj0gSSYAAAEoAAAAVmNtYXDNn+g2AAAB5AAAAGRnbHlmslg
PRQAAAnwAABvMaGvHZA6+wXwAAADQAAAAANmhoZWEHfAOBAAAArAAAACRobXR4YcP/xgAAAYAAAAAB
kbG9jYU2IVXoAAAJIAAAANG1heHABLwC3AAABCAAAACBuYW1lNl/OpQAAHkgAAAKFCg9zdBxr6o4
AACDQAAABawABAAADUv9qAFoEAP/i//0D6wABAAAAAAAAAAAAAAAAAAGQABAAAAAAQAAxGQXJ18
PPPUACwPoAAAAANXpvlcAAAAA1em+V//iAAD6wPpAAAAACAACAAAAAAAAAAAAEAAAAZAKsADAAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQPPAZAABQAAAnoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnFwNS/2oAWgPpAJYAAAAABAAAAAAAAABAAAAAPoAAA
D6AAAA+gAAAPoAAAD6AAAA+j/9wPo//AD6AAAA+gAAAPo/+ID6AAAA+gAAAPoAAAD6AAAA+gAAAP
oAAAD6AAAA+gAAAPoAAAD6AAAA+gAAAPoAAAD6P/9A+gAAAAAAAAIAAADAAAAFAADAAEAAAAUAAQ
AUAAAAAQABAABADnF//AADnAP//AAAAQAEAAAAAQACAAMABAAFAAYABwAIAAKACgALAAwADQA
OAA8AEAAARABIAEwAUABUAFgAXABgAAAAAFgAqgFGAKACjAL0A0AEGASOBPgFjgZeBrIHiggWCMI
JOAoUCpILWgwUDI4NXA3mAAIAAAAAA3sD5wANADcAAAEeARcWNicuAScmJw4BBRc3FxEUDwEOAR8
BHgE/ATUfARY7ATI2LwImPQEzMjY3NS4BIyEnNycBxQUHDNDNBwUnGwkJLTf+Qha70geWBQMDMwQ
MBtZ5HgQLYAgIAStpBc8HCAEBcAf+O5OsFgL/HcKGCEzHCgGAgEBP+4WutP+3AgFYgQMBk4FAwS
Mi3KJDAsHxGIFBs0IBz4HCZOsfGABAAAAAADhwPoAAMAFAdAC8AADcXNy8BHgEHNzEXBzYWFx4
BNwEGFiUOASImNDYyFiUGBwE+ATc+AScuAjEmBw4BAVjxVyMCAwUjgyIKGA41Tyj+rgcDAjkBK0A
rK0Ar/gkkCgFfGjEYRaNML1M2NzRekOJY8lg5DxkII4QjAwECBwIGAVInUFQgKytAKyUDMJH+oQY
WEizNfkJQJyUCB4cAAAAFAAAAAAPpA+cACwAXACMAWABhAAAlDgEHLgEnPgE3HgElDgEHLgEnPgE
3HgEFHgEXPgE3LgEnDgEDBg8BDgEfARYPASyGBw4BFx4BNz4BPwE+AS8BJjY/AjYWHwEeATczJy4
BIwciLwEuASciNx4BMjY0JiIGAUICSjk4SwEBSzg5SgJ3AU54OEsCAks4OEv+xwJmTk1nAgJnTU5
mhRENfCESEkEGCD9EstQUARMzsEUaHQF4EwYOPwQEBkwECBAFIw83IpgBASIA SQ0GKRA7IhIGATF
JMDBJMcy4SwEBSzg4SwEBS0k5SgICSjk4SwEBSzhOZwICZ05NZwICZwGbBgpeGk4mhQsLU0EXWCZ
XJ1kaQB1CJJ4aQB5/Bw4FOgIEBQ1HHIIBQB0iaQtShiUBWYQxMUKwMAAEAAAAAAPiA+cAaAB1AJ4

```

AqgAAEyIHDgEHFT4BNx4BMjY3HgEyNjceATI2Nx4BMjY3HgEyNjceARc1LgEnJiM1ByIHFAYHLgE
 1JiM1ByIHFAYHLgEnJiMnFSIHFAYHLgE1JiMnFSIHFAYHLgE1JiMnFSIHFAYHLgE1JiM1AQYWFxY
 +ASYNJiMOAQUEAR8BHGEzPgE1FBYXPgE1FBYXPgE3LgEnLgEvAS4Bjzc2NycmJw4BEx4BFz4BNy4
 BJw4BUggEASiIhigLCyk+KQsLKT4pCwsqPSoKCyO9KgsLKT0qCwspHiIjAQQIAQcFJCIjJAQIAQc
 FJCIjIwEECAEIBCQiIyQECAEIBCQjIiQECAEIBCQjIiQFCAELDTA0NE4fJzMTEy1C/t0NfB4LCh0
 TKikqKikqKSoaJAoMaxMOTCpQMHQFBgYBCS0fECg4ATosLDoBATosLDoBGgYDIwMrArkMDRGYDQ0
 YGA0NGBgNDRGYDQ0YGA0MGQErAyIEBgEBBgMkAgIkAwYBAQYDJAICiWQGAQEGAYQCAiQDBgEBBgM
 kAgIkAwYBAQYDJAICJAMGAQGCNfKRDtZnVBAFATGUHII2JgkMAyoDAyoDAyoDAyoDARULDMkNBg8
 GDA5SBTo2MgEFHzOPAScsOgEBOiwsOgEBOgAAAAAAMAAAAA+ID5wALABcAKQAAARY2NxY2Ny4BJxY
 GAX4BFz4BNy4BJw4BBQclFwUWNjcWNjceATcWNjceBAsgqYQqGUXksRBwCotABQzIyQwEBQzIyQ/3
 ADwE6Lv66G2wkI10eGF8oJ1kb/sACgjYaBhUJwMTHwhNAQYyQwEBQzIyQwEBQzU7SjukSQIuNAw
 rKhI8OxUgAWsAAv/3AAAEgPnAAwAQQAAYWFxY2NzYmJyYASUGFhcWNjceXAwYfAgcGFB8BFjY
 /ARcBhY7ATY/Asc2LwE3FRQWOWEyNjURNcclLgEnJyYCYBYXJy1TGRYXJxs6MP28Fis3NmYaYJQ
 DAgN1fAQFRgYMBHECHQEJB18MAYkBAgVJQAKGPgcJCP6kF1ssIDMDhihUGBYWKChUGA8BHBgpYSE
 eCSU3/wAGBwZ1jwUMBT0EAQSCARUHCwEM+AQGB1VvdAYJCQYBGQkFySc0AQEYAAL/8AAAAxID5wA
 DACoAACUXJSsCBjgYXPgEXHgEjLgEHIGYXHgEXJSuAQcmNjceARc3NgInLgEnJgYBI2MBjWb+EA2
 eCgaHOQMBIAJcQwguOQaTQAFIA2uNag8KAXt9EGAHGn8BcWIGr9vWtNMCogOVxgcNVQIaCUkqVU8
 GdCSbBpU7BwQaAQUNs8OATyABUsGAAAAAABAAAAAADtwPnAEwAUACJAJIAABMGJxU2Nz4BFzY
 XHGzByYPAQ4BFBYXfjI/AT4BJz8BNhc2FjI3PgEzNhYyNz4BMzYXHgEzNSImJyYHJgYiJy4BIyY
 GIicmIgYiJgciAQcjPwEvASYHBg8BFSEVMjY3NjcyFhcWMjc2MhceATMyNzYyFyYzYmjc2NxY2Nz4
 BNzUnITcXfjY3NjcyY3NjcyATI2NCYiBkscGxscDh4OHxsKFW1GJiFMCgkJCRITeVAOBALfExsfHDC
 3HA4eDhw3NxxOHg4GgW81EBA1DxsGzC4Gw4fDh82OBsXQDc3NiAbAgN/GTghyBALHSAIOv7oERc
 LICUSJRAULxghTCILFQsUGCFNIRQYExggJQsTBww1Ezr+1XkMEhsFBAXcF/6NAS9ILy9ILwFzDQF
 ZAQ4HBgEBDQUHShcgVgcVFxYHEhJTDiMTYgkNAQEbdGcFARsOBwUBDQcKWQcHdQEBGg0HQBQeADQ0
 aGgEBH4asFk4JChMSINcHeQQFDwEHCQkJDw8FBAkPDwkJDwEBBwkEDgszAXwFCRINFhFeE1kkLy9
 ILy8ABAAAAAAD4gPnAAMABwA8AEkAACUzNSMFMzUjAQcxBxUnJgYPAQYWHwIHBg8BDGefAR4BPwE
 2PwEXFh8BFj8BNiYvASYvAQm3PgEvASYjIiUOARceATc+AScuAgMou7v82bq6ApGt15UHCgIPAgc
 G7z5NBQOABQEEQgQNBX8EBIqLBgafCwYtAwYHkgYEQpmsBgMDHwUJBP49JxYWGfQoKBYWDzE5JJy
 cnALgZHwBJgEGBjwHCwE9aiACA2sEDQVTBQIEZgMCOGAEEASICC2MHDAIpaQMxAQhkBAwFNgg3GVM
 pJxcWGVmpGhwBAAAF/+IAAAPrA+cACAARAC0APwBHAAABDgEiJjQ2MhY3FAYiJjQ2MhY1FTMyFhQ
 GKwEVFAYiJj0BiYImNDY7ATU0NjIWBxYXFjY3Mx4BNzY3NiYnIQ4BARUzNS4BIgYCY5QETHhMTHN
 0FB0UFB0U/ZBGCg0NckYNEw5FCg0NckUOEw3GFYd4vxR3Fb54JxdBe3v94nx7Aec+AREbEQHTDxM
 THhQUZQ8UFB4TFBRFDRQNRQoNDQpFDRQNRQoNDfAnFjxjbW1jPBYNsgsFBcgBr8PDCw4OAAAYAAA
 AA+ID5wALABcAIwAvAFQAXQAAAQ4BBY4BJz4BNx4BBQ4BBY4BJz4BNx4BFx4BFz4BNy4BJw4BBR4
 BFz4BNy4BJw4BAQcGFRQWHwEVHgE3PgE9ATYmLwE3Fx4BOWE2NCcjJzQmJyMiBjceATI2NCYiBgO
 rAlU+QVMBAVU/QFP9ugJVPkBUAQJVPj5V6AJ1V1hzAgN0V1Z1/bYCC1hYcwICdVZYcwGXqhASD60
 BIBgQFAIHGJpNAGRCJMiIoFXGhsDBhxAASxBLcXBLAF2QVMBAVU/QFMCA1NAQVMBAVU/QFMCA1N
 AWHMCAnVWHMCAnNYVnMEAnVWHMCAnMBjp4UGBIeCFqnFhgDAXkRyAILETRpTwoJAzcefwIYAgc
 yISwsQswsAAAAAUAaaaaa+ID5wBBAEWAVQCEAI0AADcyFhceARcxMzE+ATc+ATczMTIWFx4BFz4
 BNz4BMhYXHgEXNSImJy4BJw4BBw4BIiYnLgEnIzUOAQcOASImJy4BJxMGfH8BNycmByIGJR4BMjY
 0JiIGJQC0AQcGFj8BFwcGDwEjIgYUFhchMjY/AhcWFzM+ATQmJyMvASYxLwEzJyYnIhceATI2NCY
 iBgESGBETPDafMTwTDxgPBBMaERQ/NDRAExIZJRORFEA0FhwSEz0xMT0UER0qHBITOi4HMj8UEX0
 pHRMTPzMBGw8UziXOCgoOGQM+ASK/Kio/Kf451Q8YAgIkGn9IeAcEptUUGhoUAQMNfGZYgFQOEQA
 UGhoUg24pAQMFAn0OEABKASo+Kio+KqMQEHUpAgIpFREQARESFsKCAigWEhAQEHUpAmISEhQoAQE
 nFRISEhIUJgMBAigWEhISEhYpAQHNEYQJULxSBAEQWYApKT8qKlsdBBgQGyAEGTd4BwqbGigaAQ4
 MeIBLDAEBGigaAXMiAQIDYgoBGR8qKj8pKQAAwAAAAADCQpNAAQACQAXAAABFgIHEQMMAj8CNSM
 VJgADPqEXBzEhHgEXIT4BNyE1PgEXEAInNRY2JwYmNT4BJwYmAZ5bRhwhfHfTxBh8fGf6xAwfZghb
 +qA9gSwGUTWsC/o4Fqmn7HTsqARobJhECKTDZav+qS0CNf1VMAGh0QlqEHcC/rf+nBNxhwM/XR4
 mjAh2FoubASkBCYmCCsEGggCFCMBJwOABgAAAAAD4gPnACQARwBTAf0AfACWAAATBiMVMjYyFjI
 2MhYyNjIWMjYyFjM1IiYiBiImIgYiJiIGIiYiEwYiJyYnFRYXFjI2MhYyNjIWMjYyMhclJiMiBiI
 mIgYiJiIFFBYXPgE3LgEnDgElFBYXPgE0JiIGJQUOAR8CBRYXFjI2MhYyNjIWFzI3AzC+ATUuASc
 iJQC0AR8CBxUWFxYyNjIWMzI3Jzc+ATQmIz4cISE5RDpCOUQ5QjLEOkE6RDkhITLEOkE6RD1COUQ
 5QjpEARIpEgcHBwcSKSQqJCKkKiQpJBYPDg4PFiQpJCokKSQqAm9BNTJDAQFDMjJD/fYoIh8qKj8
 qAlP+nhUfDQnc/qUODB1COUQ5QjLEOSEYFsXlGiABJhWE/dfdDRMHAjPHDgWKSksMrJBQPDnuPEBQ
 YEGEDV0bGxsbGxsbG10bGxsbGxsbAUICAMCOgIDCBERERERBTOfEREREU4zQwICQzQzQwICQ7k
 gKgEBKkAqKkNBbTAjB6nBBAUNghoaGgEIAWcrBSQaHSYBbikDHhYEaidQAgUIEREE4RoEfIMYAAA
 AAwAAAAADCAPnABIATQBbAAATDwIGFh8BFj8DNI8BJiMiEwYxBwYWHwEWNj8BFwMGHwEHBhYfARY

2PwE2JyYvATcXMDEXFjY/ATYmLwEmBg8BJyImLwEiMSYjJwY3BhYXFjY3NiYnJiMiBpc9AlQEAWd
UCwhhAigECEgFBgkuAUMCBAY5BgwDLcRzBAi1PwMFB1UGDAJqAwUBAlNaLDAFDQSCBQIFMAUMBFE
+AQICxAEDA4gJyRQkMDFcFxQkMBgYJT4Ba4oEawYOBQCFCoMDWwsHQgQBpgGXBgWdGQIEBmUD/vw
LCKiPBwsDJQMEB+8IBwMCTc0lKAQBBZwFDQqoBAEFYTUBAVcCDAF7MVwXFCQwMVwXciYAAAMAAAA
AA8MD5wBRAFOAcAAANw4BBxUyNjc+ATM2Fx4BMj4CMzYXHGEzFjc+ATM2Fx4BMxY3PgEzNhceATM
1IicuASMMBw4BBYInLgEjJgcOAQciJy4BIYyHDgEHIicuASIGEx4BMjY0JiIGAQMHBgcDBh4BNj8
BJTY3EzYmJy4BBjwOHRAQHAWOHxEhHQ0dIRwcIBAhHQ4dEB8cDh8RIRwOHRAgGw4gECEdDh0QIBs
OIBAhHQ4dEB8cDh8QIhwOHRAgGw4gECEdDhwRIBcOICEfggEuQy8vQy4Cf6XyEgq6CQghIwqDAR4
SCsMMDBUOHRtsBQcBWQcFBQcBDQUHBwOHAQ0FBwENBQcBDQUHAQ0FBwENBQdZDQUHAQ0FBwENBQc
BDQUHAQ0FBwENBQcBDQUHBwGLIS4uQy4uAa3+6ZiJE/69ESMUCRDjsQkTAU8WLAwFAQ4AAAAAAwA
AAAA4gPnACQAMABOAAATBiMVMjYyFjI2MhYyNjIWMjYyFjM1IiYiBiImIgYiJiIGIiYiARQWFz4
BNy4BJw4BEwUOAR8CBRYXFjI+ARYyPgEWFzI3AzC+ATUuASc+HCEhOUQ6QjLEOUI5RD1COU5ISE
5RT1COUQ5QjLEOUI6RAKCQTUyQwEBQzIyQ0n+nhUfDQnc/qUODB1COUQ5QjLEOSEYFsXlGfABJhw
BhA1dGxsbGxsbGxtDgXsbGxsbGwEGM0MCAkMzM0MCAkMBHEEFMCMHqcEDBg0aARsaARsBCAFnKwU
kGh0mAQAEAAAAAONA+cAEABQAGUAKQAAJQcxBgCGHgI/ATY0LwEmIjCHfzCXBxc3FwcXNxcHFzc
2JiMuAQcOAQcnNz4BMzYmIyYGBw4BBYc3PgEXNiYjJgYVDgEXByc+ATM2JjUmBgCBHgEHDgEHDgE
nLgE3PgE3PgE3MhYFDgEHBhcPAicmIg8BBhQfARYyPwE2NC8BPwEWMzI2Nz4BNzYmJy4BIYIGAZM
fBAIPBiguECUDA00ECVdDDgkMCwsPCg8NEQsSDJALAgEMJAIiIwMLCBQrAhUEAgshAh8cAwkBEC0
CEAICDRcoFgEBCw8vAgEJCQsDATQXEWfKSIJGoDUXEWfKSIpXi0eNf7NKTiHCiYLCLOfChkJjwk
JJAoZCY8JCQXAMjI9OnQyKTIHbX0hHUsqOXbgHQIDDi8qCg4iAwkDVAOVkBAUDREMDgsODgsLCg0
6Bg4MAQEKJgUMBBoUFRAFCAEMKAgKAh8eARYQBACBFjACAQwnIgwXAQ0BBAIGF0InKk8hRBgxFOI
nKk8iKCKBFBIPyJRTPiUWuwUJCY8JGQokCQmPChgKBb8SHTIyKWE0NlwiHRwyAAUAAAAAA+ID6QA
DAACACwATAEwAADchNSEnITUhJyE1ISUOAQcXNQYmAQYWFwUeARCFDgEfAR4BPwEVDgEHLgE3PgE
3NhYXNZuUAScOAQceARc+ATc1NzMWnXElLgEHIGY3fQEY/ug+ARj+6D4BN/7JA3gEFgKGKzv93zI
fBgE7WmAT/uwPDQUDBhkPbgU+LTFAAQEFgCY/GmcDZzlefAMDF5dfwpBCS8i/ssjOQ8FBwK9Hx8
fHx85J0MCCnkIBAJFFjgDriRiK0sFGRANEBAEHworOAEbQjEcMA8WDBMeCAZVBgN9XV19AgJ9XRM
dAQ4Bfis7JAICagAAAAYAAAAAA+ID5wAoAFEAWgBjAhCAiWAAExYXHgMyPgIyHgIyPgI3NjQnDgM
iLgIiDgIiLgInBicWFzIeAjI+AjIeAjI+AjM2NCciDgIiLgIiDgIiLgIjBiUeATI2NCYiBgUeATI
2NCYiBiUHBh8BFjMyNz4BLwE3NiYnJiMGJQCgHwEWMzI3PgEvATc2JicmIyIBAQsqNs6XTksNFU
0LDpdOSw0KwsLLzorNFU0LDpdOSw0VTQsOS8LAQELKjUrOl05LDRVNCw6XTksNCsLCy86KzRVNCw
6XTksNFU0LDKvCwLYAS1DLsXELf31ATBILY9IMAGwUAKUgwTCgkNBwhHRgcIDQkJFP31VgoLWAw
VCgoOBwhMSwcJDgkJFgG4CwEBEBcSEhcQEBCSEhcQAQEWAQESFxAQFxFISFxAQFxiBAWkLARAXEhI
XEBAXEhIXEAEWARIXEBAXEhIXEBAXEGf5IS0tQy0tHSMwMEcwMOSVEhKHEQUJHA11gg4cBwUBBJ4
TE5ASBQkeDn2KDx0JBAAGAAAAAaOBA+cADwAoAD8AUgBiAHUAAAEGBw4BBx4BMz4BNy4BJwYlBgC
GBYMiJiMeARc+ATc+ATcmJy4BJw4BExQXHgEXHgEXFhc+ATU0Jy4BJy4BJyIFFBYXNTIWPwEuASc
mNz4BNw4BJQYHPgEXHgEXFhcUAScOAScOAhYXPgE3JicmNjc+ATcOAQI6UGkuUSAzcD12wD0ZOR0
z/q5HUBcWJxUgCxVNMhdvRk+nNSonLkIOH0pWIRBCLCJBG0IuGRwID0EuJmtBM/4LBggBVEIZERU
HDAQCCARGUAHDBQEqVywoUig4Ii7PiAUT+gIPChMnMWgwGQgEAgUEEQdCcwEQMR0MDwMcHwFpWSI
0FUx2GACaQNNbCoBDxQUXFECBUWAhclARFWVQXFXBEjFS43L2o5KywLiXIPFwG+GTEVCAUFAYB
EKETXiJwXP6vGISoIAGUEEQ8VFXygDA1QFQ5XfJ5JciYhRFAMtygwSxMCJQACAAAAAANUA+cACAB
OAAATHgEyNjQmIgYlASc+ATUuAScOAR0BiGYHbHqfARYyPwI2PwEWHwEWHwEOAQcRHgEyNjc1NwE
eATc+AS8BNzYmLwEmLwI3NjQnLgEjInsBNlE2NlE2AZf+kCkMDwEbFhIaDBQHDw9tESkQBGMEBUw
JEjUBBT4JiGIBGigaARCBAA0iDxQCEfQMHWckgQCIDgaqEBAHEwkSAz8pNjZRNzdx/o4nBhYQFB0
BARYQBgyJECKRbQ8PCQMBBU0SETMEBUYRPAB+7RQaHhX7JP8ADQKQDzAS8wwjWCF4BwUJA60QJw8
HCAAAB//9AAAD6APnABYAJQAzAEIAUQB2AIIAADcGBw4BBxUFLgEnLgUnLgEjJgYBMhceAQcOASc
uATc+ATcHBhYXFjY3NiYnJiMOASUWFx4BBw4BJy4BNz4BNycOAQcGFhcWNjc2JicmIyUHBgcGFhN
BBwYeATY/AT4BLwE3Fx4BHwEWNi8CNcyvASYiBjcGHgE+ASYNJiMiBpY3PQwUAQPFCLFGP0YBjK
CHxg7Ix9DalkPDz4/DQ9mPT4/DQ10M8URWFRXiRURWFQXF0du/oAPDj1ADBB1PT8+DA1PMgNHARE
RV1VXiHURWVMXFWFGVRMGBAoMiiUEEiUdBS4CAhdQdyAFDgiIIAwfeTMSGAMDDxNVBiA9Mg4gHgc
IGirKIBMBwN8BgVEBAImBBcQGQ8LEQEQAQRMDEGQ/Pj8NEGU9NT4BaFaKFRFYVFeJFQUBVqABAw9
kPz4/DBB1PTU+AjgDVUUhIihcRWRWihUF3XgPFxMhCnukExwJEhPOASQVRUtTCw0BHwQzCxuOAh0
HAQEEIR4zDiA9MwgBHwAAAAwAAAAAA8MD5wADAACACwAPABMAFwAbAB8AIwAnAEoAVwAAJTM1Iwc
zNSMHMzUjBzM1IwcZNSMHMzUjBTM1IwUzNSMFMzUjBTM1IwEHBhYXFhcWMYUXFj4BJi8BBzcXNRU
3PgEuAQ8BLwEmIyYGNwYWFxY2NzYmJyYOAQNIHx9eHx/ZHx9dHx/aICBdHx8DCT4+/ug+Pv7KPj7
+yT4+ARiBGRktGxwJCAFnPR0zFBoc3r1DZKcbIAMkGoPdBBMTGSt9FhcnKVMZFhcnGzowJD8/Pz9
BQUFBQUFBQZycnJycnJwCJN4vXxwPAQEZOAKZojMLS11HwEBCgIlNCEBB0MBCAEYtIlTGRYXJy1
TGQ8CGwAAAAAAAEgDeAAEAAAAAAAAAAQAAAAEAAAAAAAAEADwABAAEAAAAAAAAAIABwAQAAEAAAAAAM

```
ADwAXAAEAAAAAAAAQADwAmAAEAAAAAAAAUACwA1AAEAAAAAAAAAYADwBAAAEAAAAAAAAoALABPAAEAAAA
AAAsAEgB7AAMAAQQJAAAAAgCNAAMAAQQJAAEAHgCPAAMAAQQJAAIADgCtAAMAAQQJAAAMAHgC7AAM
AAQQJAAQAHgDZAAMAAQQJAAUAFgD3AAMAAQQJAAAYAHgENAAMAAQQJAAoAWAERAAAMAAQQJAAAsAJAG
DIEFjY29yZGlvd19JY29uc1JlZ3VsYXJBY2NvcnRpb25fSWNVbnNBY2NvcnRpb25fSWNVbnNWZXJ
zaW9uIDEuMEFjY29yZGlvd19JY29uc0ZvbnQgZ2VuZXJhdGVkIHVzaW5nIFN5bmNmdXNpb24gTWV
0cm8gU3RlZGlvd3d3LnN5bmNmdXNpb24uY29tACAAQQBjAGMABwByAGQAaQBvAG4AXwBJAGMABwB
uAHMAUGBlAGcAdQBsaGEAcgBBAGMAYwBvAHIAZABpAG8AbgBfAEkAYwBvAG4AcwBBAGMAYwBvAHIA
ZABpAG8AbgBfAEkAYwBvAG4AcwBWAGUAcgBzAGkAbwBuACAAMQAuADAAQQBjAGMABwByAGQAaQB
vAG4AXwBJAGMABwBuAHMARGBvAG4AdAAgAGcAZQBwAGUAcgBhAHQAZQBkACAAdQBzAGkAbgBnACA
AUwB5AG4AYwBmAHUAcwBpAG8AbgAgAE0AZQB0AHIAbwAgAFMAdAB1AGQAaQBvAHcAdwB3AC4AcwB
5AG4AYwBmAHUAcwBpAG8AbgAuAGMABwBtAAAAAIAAAAAAAAAACgAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAGQECAQMBAEFAQYBBwEIAQkBCgELAQwBDQEOAQ8BEAERARIBEWUARUBFgEXARgBGQEaAA1
KYYZ1bGluX1Rocm93DFRhYmxlX1Rlbm5pcwtDeWNsaW5nX0JNWApYXRlc19Qb2xvCFN3aW1taW5
nDERpc2N1c19UaHJvdwZCb3hpbmcGUm93aW5nCUhpZ2hfSnVtcAxiJmRvb3JfR2FtZXMKQ31jbGV
fUmFjZQtYXRlc19HYW1lcwDTYWlsaW5nEU1hcmF0aG9uX1N3aW1taW5nCE1hcmF0aG9uBkRpdml
uZW1Td21tbWluZzEJQmFkbWludG9uB1JhY2luZxVTeW5jaHJvbm16ZWRFU3dpbW1pbmcLVm9sbGV
5X0JhbGwJQXRobGV0aWNzFEN5Y2xpbnRmdfTW91bnRhaW5CaWt1CUxvbnRmdfSnVtcAAAAA==)
format('trueType');
    font-weight: normal;
    font-style: normal;
}
.e-acrdn-icons {
    font-family: 'acrdn-icons';
    font-size: 16px;
}
.cycle_BMX::before {
    content: "\e702"
}
.javelin::before {
    content: "\e700";
}
.marathon::before {
    content: "\e70e";
}
.tennis::before {
    content: "\e701";
}
.waterpolo::before {
    content: "\e703";
}
.swimming::before {
    content: "\e704";
    position: relative;
    top: 5px;
}
.discus::before {
    content: "\e705";
}
.boxing::before {
    content: "\e706";
}
.rowing::before {
    content: "\e707";
}
.highjump::before {
    content: "\e708";
}
```

```
.cycle::before {
    content: "\e70a";
}
.sailing::before {
    content: "\e70c";
}
.marathan_swim::before {
    content: "\e70d";
}
.boxing::before {
    content: "\e706";
}
.dive::before {
    content: "\e70f";
}
.swimming_In::before {
    content: "\e710";
    position: relative;
    top: 2px;
}
.badminton::before {
    content: "\e711";
}
.sync_swim::before {
    content: "\e713";
    position: relative;
    top: 3px;
}
.volleyball::before {
    content: "\e714";
}
.cycle_Mountain::before {
    content: "\e716";
}
.longjump::before {
    content: "\e717";
}
.e-athletics::before {
    content: "\e715";
}
.e-water-game::before {
    content: "\e70b";
}
.e-racing-games::before {
    content: "\e712";
}
.e-indoor-games::before {
    content: "\e709";
}
.e-acrdn-icons:not(.e-icons) {
    padding: 0 16px 0 0;
    vertical-align: middle;
}
#athletics li,
#racing_games li,
#water_games li,
#indoor_games li {
```

```

        line-height: 36px;
        list-style-type: none;
        text-indent: 16px;
    }
    .accordion-control-section {
        margin: 0 10% 0 10%;
    }
    @media screen and (max-width: 768px) {
        .accordion-control-section {
            margin: 0;
        }
    }
</style>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Add font awesome in ##Platform_Name## Accordion control

We can customize the Accordion component items by using font awesome icons.

- Need to add font awesome CDN reference link in your sample to use font awesome icons.

```

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css" />

```

You can add the font awesome icons to the Accordion header using '[iconCss](#)' property. The following sample illustrates how to use font awesome in Accordion component.

INDEX.JS

```

ej.base.enableRipple(true);
//Initialize Accordion component
var accordion = new ej.navigations.Accordion({
    items: [
        { header: 'Twitter', expanded: 'true', iconCss: 'fa fa-twitter',
        content: 'Twitter is an online social networking service that enables users
        to send and read short 140-character messages called "tweets".' },
        { header: 'Facebook', iconCss: 'fa fa-facebook', content: 'Facebook
        is an online social networking service headquartered in Menlo Park,
        California. Its website was launched on February 4, 2004, by Mark Zuckerberg
        with his Harvard College roommates.' },
        { header: 'WhatsApp', iconCss: 'fa fa-whatsapp', content: 'WhatsApp
        is an American freeware, cross-platform messaging and Voice over IP (VoIP)
        service owned by Facebook, Inc.' },
    ]
});
//Render initialized Accordion component

```



```
accordion.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Accordion</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Toolbar Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
>
  <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-
awesome.min.css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
    <br><br>
    <div id="result"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Customization

Use following CSS to customize the accordion items header icons.

```
`css
.e-accordion .e-acrdn-item .e-acrdn-header .e-acrdn-header-icon {
display: inline-block;
padding: 0 10px 0 0 !important;
}
`
```

Use following CSS to customize the selected accordion item content icons.

```
`css
.e-accordion .e-acrdn-item.e-select .e-acrdn-panel .e-acrdn-content .e-content-icon {
color: rgba(0, 0, 0, 0.54) !important;
}
`
```

Customize expand collapse actions in ##Platform_Name## Accordion control

Accordion component supports customizing the expand or collapse animation action behavior. You can manually change the expand animation action performed after the collapse animation operation performed on already expand pane when the expand icons are clicked.

By default, the Accordion component pane is expanded or collapsed, when click the expand or collapse icon. It is not affected on already expand pane.

The following sample demonstrates, how to expand the collapsed Accordion item after collapse animation performed on the expanded Accordion item using [created](#), [expanding](#), and [expanded](#) event. In the Expanding event, get the previously expanded item index and prevent the expanding behavior using `args.cancel` option. Expand the Accordion item dynamically based on specifying the `index` value using the [expandItem](#) public method and [expanded](#) event.

INDEX.JS

```
ej.base.enableRipple(true);
//Initialize Accordion component
var expandIndex;
var isCollapsed = false;
var initialLoad = true;
var accordion = new ej.navigations.Accordion({
  expandMode: 'Single',
  expanding: onExpanding,
  expanded: onExpanded,
  created: onCreate,
  items: [
    { header: 'ASP.NET', expanded: 'true', content: 'Microsoft ASP.NET is a set of technologies in the Microsoft .NET Framework for building Web applications and XML Web services. ASP.NET pages execute on the server and generate markup such as HTML, WML, or XML that is sent to a desktop or mobile browser. ASP.NET pages use a compiled, event-driven programming model that improves performance and enables the separation of application logic and user interface.' },
    { header: 'ASP.NET MVC', content: 'The Model-View-Controller (MVC) architectural pattern separates an application into three main components: the model, the view, and the controller. The ASP.NET MVC framework provides an alternative to the ASP.NET Web Forms pattern for creating Web applications. The ASP.NET MVC framework is a lightweight, highly testable presentation framework that (as with Web Forms-based applications) is integrated with existing ASP.NET features, such as master pages and membership-based authentication.' },
    { header: 'JavaScript', content: 'JavaScript (JS) is an interpreted computer programming language. It was originally implemented as part of web browsers so that client-side scripts could interact with the user, control the browser, communicate asynchronously, and alter the document content that
```

```

was displayed. More recently, however, it has become common in both game
development and the creation of desktop applications.' },
]
});
//Render initialized Accordion component
accordion.appendTo('#element');
function onCreate() {
    initialLoad = false;
}
function onExpanding(e) {
    if (e.isExpanded && !initialLoad && !isCollapsed) {
        e.cancel = true;
        expandIndex = accordion.items.indexOf (e.item);
        isCollapsed = true;
    }
}
function onExpanded(e) {
    if (!e.isExpanded && !initialLoad && isCollapsed) {
        accordion.expandItem(true, expandIndex);
        isCollapsed = false;
    }
}
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Accordion</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Toolbar Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
>

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
        <br><br>
        <div id="result"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";

```

```

}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Integrate treeview inside the accordion in ##Platform_Name## Accordion control

Accordion supports to render other Essential JS 2 Components by using content property. You can give content as an element string like below, for initializing the component.

```
`js
```

```
content: '<div id="element"> </div>'
```

```
,
```

The other component can be rendered with the use of provided events, such as [clicked](#) and [expanding](#).

The following procedure is to render a TreeView within the Accordion,

- Import the **TreeView** module from **ej2-navigations**, for adding TreeView. Please refer the [TreeView initialization steps](#)
- You can initialize the TreeView component in [expanding](#) event, by getting the element and defining the required TreeView properties.

INDEX.JS

```

ej.base.enableRipple(true);
var DocDB = [
  {
    nodeId: '03', nodeText: 'Documents', icon: 'folder',
    nodeChild: [
      { nodeId: '03-01', nodeText: 'Environment Pollution.docx',
        icon: 'docx' },
      { nodeId: '03-02', nodeText: 'Global Water, Sanitation, &
        Hygiene.docx', icon: 'docx' },
      { nodeId: '03-03', nodeText: 'Global Warming.ppt', icon:
        'ppt' },
      { nodeId: '03-04', nodeText: 'Social Network.pdf', icon:
        'pdf' },
      { nodeId: '03-05', nodeText: 'Youth Empowerment.pdf', icon:
        'pdf' },
    ]
  },
]

var DownloadDB = [
  {
    nodeId: '05', nodeText: 'Downloads', icon: 'folder',
    nodeChild: [
      { nodeId: '05-01', nodeText: 'UI-Guide.pdf', icon: 'pdf' },
      { nodeId: '05-02', nodeText: 'Tutorials.zip', icon: 'zip' },
      { nodeId: '05-03', nodeText: 'Game.exe', icon: 'exe' },
      { nodeId: '05-04', nodeText: 'TypeScript.7z', icon: 'zip' },
    ]
  }
]

```

```

var PicDB = [
    {
        nodeId: '04', nodeText: 'Pictures', icon: 'folder', expanded:
true,
        nodeChild: [
            {
                nodeId: '04-01', nodeText: 'Camera Roll', icon:
'folder', expanded: true,
                nodeChild: [
                    { nodeId: '04-01-01', nodeText:
'WIN_20160726_094117.JPG', image:
'http://npmci.syncfusion.com/development/demos/src/images/employees/9.png'
},
                    { nodeId: '04-01-02', nodeText:
'WIN_20160726_094118.JPG', image:
'http://npmci.syncfusion.com/development/demos/src/images/employees/3.png'
},
                ]
            },
            { nodeId: '04-02', nodeText: 'Wind.jpg', icon: 'images' },
            { nodeId: '04-03', nodeText: 'Stone.jpg', icon: 'images' },
        ]
    }
]
//Initialize Accordion component
var accordion = new ej.navigations.Accordion({
    expanding: expand,
    width: 400,
    items : [
        { header: 'Documents', expanded: true, content: '<div
id="treeDoc"></div>' },
        { header: 'Downloads', content : '<div id="treeDownload"></div>'
},
        { header: 'Pictures', content: '<div id="treePic"></div>' },
    ]
});
//Render initialized Accordion component
accordion.appendTo('#element');
function expand(e) {
    if (e.name === 'expanding' && [].indexOf.call(this.items, e.item) === 0 &&
e.element.querySelector('#treeDoc').childElementCount === 0) {
        let treeObj = new ej.navigations.TreeView({
            fields: { dataSource: DocDB, id: 'nodeId', text: 'nodeText', child:
'nodeChild', iconCss: 'icon', imageUrl: 'image' },
            sortOrder: 'Ascending'
        });
        treeObj.appendTo('#treeDoc');
    }
    if (e.name === 'expanding' && [].indexOf.call(this.items, e.item) === 1
&& e.element.querySelector('#treeDownload').childElementCount === 0) {
        let treeObj = new ej.navigations.TreeView({
            fields: { dataSource: DownloadDB, id: 'nodeId', text: 'nodeText',
child: 'nodeChild', iconCss: 'icon', imageUrl: 'image' },
            sortOrder: 'Ascending'
        });
        treeObj.appendTo('#treeDownload');
    }
}

```

```

    }

    if (e.name === 'expanding' && [].indexOf.call(this.items, e.item) ===
2 && e.element.querySelector('#treePic').childElementCount === 0) {
        let treeObj = new ej.navigations.TreeView({
            fields: { dataSource: PicDB, id: 'nodeId', text: 'nodeText', child:
'nodeChild', iconCss: 'icon', imageUrl: 'image' },
            sortOrder: 'Ascending'
        });
        treeObj.appendTo('#treePic');
    }
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Accordion</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Toolbar Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
        <br><br>
        <div id="result"></div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

AccumulationChart

Pie dough nut in ##Platform_Name## Accumulation chart control

Pie Chart

To render a pie series, use the series [type](#) as `Pie` and inject the `PieSeries` module using `AccumulationChart.Inject(PieSeries)` method. If the `PieSeries` module is not injected, this module will be loaded by default.

INDEX.JS

```
var piechart = new ej.charts.AccumulationChart({
  series: [
    {
      dataSource: [{ x: 'Jan', y: 3 }, { x: 'Feb', y: 3.5 }, { x:
'Mar', y: 7 }, { x: 'Apr', y: 13.5 },
      { x: 'May', y: 19 }, { x: 'Jun', y: 23.5 }, { x: 'Jul', y: 26 },
{ x: 'Aug', y: 25 },
      { x: 'Sep', y: 21 }, { x: 'Oct', y: 15 }],
      radius: '100%',
      xName: 'x',
      yName: 'y'
    }
  ],
  tooltip:{enable: true}
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Radius Customization

By default, radius of the pie series will be 80% of the size (minimum of chart width and height). You can customize this using [radius](#) property of the series.

INDEX.JS

```
var piechart = new ej.charts.AccumulationChart({
  series: [
    {
      dataSource: [{ x: 'Jan', y: 3 }, { x: 'Feb', y: 3.5 }, { x:
'Mar', y: 7 }, { x: 'Apr', y: 13.5 },
      { x: 'May', y: 19 }, { x: 'Jun', y: 23.5 }, { x: 'Jul', y: 26 },
{ x: 'Aug', y: 25 },
      { x: 'Sep', y: 21 }, { x: 'Oct', y: 15 }],
      radius: '100%',
      xName: 'x',
      yName: 'y'
    }
  ],
  tooltip: { enable: true }
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```


Pie Center

The center position of the pie can be changed by Center X and Center Y. By default, center value of the pie series x and y is 50%. You can customize this using [center](#) property of the series.

INDEX.JS

```
var piechart = new ej.charts.AccumulationChart({
  series: [
    {
      dataSource: [{ x: 'Jan', y: 3 }, { x: 'Feb', y: 3.5 }, { x:
'Mar', y: 7 }, { x: 'Apr', y: 13.5 },
      { x: 'May', y: 19 }, { x: 'Jun', y: 23.5 }, { x: 'Jul', y: 26 },
{ x: 'Aug', y: 25 },
      { x: 'Sep', y: 21 }, { x: 'Oct', y: 15 }],
      radius: '100%',
      xName: 'x',
      yName: 'y',
      enter: {x: '60%', y: '60%'},
    }
  ],
  tooltip:{enable: true}
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Various Radius Pie Chart

You can use radius mapping to render the slice with different [radius](#) pie and also use [border](#), [fill](#) properties to customize the point. `dataLabel` is used to represent individual data and its value.

INDEX.JS

```
var piechart = new ej.charts.AccumulationChart({
  series: [
    {
      dataSource: [
        { x: 'Argentina', y: 505370, r: '100' },
        { x: 'Belgium', y: 551500, r: '118.7' },
        { x: 'Cuba', y: 312685, r: '124.6' },
        { x: 'Dominican Republic', y: 350000, r: '137.5' },
        { x: 'Egypt', y: 301000, r: '150.8' },
        { x: 'Kazakhstan', y: 300000, r: '155.5' },
        { x: 'Somalia', y: 357022, r: '160.6' }
      ],
      dataLabel: {
        visible: true, position: 'Outside',
        name: 'x'
      },
      radius: 'r', xName: 'x',
      yName: 'y', innerRadius: '20%'
    },
  ],
  enableSmartLabels: true,
  legendSettings: {
    visible: true,
  },
  enableAnimation: true,
  title: 'Countries compared by population density and total area'
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
</body>
</html>
```

```

var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Doughnut Chart

To achieve a doughnut in pie series, customize the [innerRadius](#) property of the series. By setting value greater than 0%, a doughnut will appear. The `innerRadius` property takes value from 0% to 100% of the pie radius.

INDEX.JS

```

var piechart = new ej.charts.AccumulationChart({
    series: [
        {
            dataSource: [{ x: 'Jan', y: 3 }, { x: 'Feb', y: 3.5 }, { x:
'Mar', y: 7 }, { x: 'Apr', y: 13.5 },
            { x: 'May', y: 19 }, { x: 'Jun', y: 23.5 }, { x: 'Jul', y: 26 },
{ x: 'Aug', y: 25 }],
            innerRadius: '40%',
            xName: 'x',
            yName: 'y'
        }
    ]
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}

```

```

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Start and End angles

You can customize the start and end angle of the pie series using the [startAngle](#) and [endAngle](#) properties. The default value of `startAngle` is 0 degree, and `endAngle` is 360 degrees. By customizing this, you can achieve a semi pie series.

INDEX.JS

```

var piechart = new ej.charts.AccumulationChart({
  series: [
    {
      dataSource: [
        { x: 'Jan', y: 3, text: 'Jan: 3' }, { x: 'Feb', y: 3.5,
text: 'Feb: 3.5' },
        { x: 'Mar', y: 7, text: 'Mar: 7' }, { x: 'Apr', y: 13.5,
text: 'Apr: 13.5' },
        { x: 'May', y: 19, text: 'May: 19' }, { x: 'Jun', y: 23.5,
text: 'Jun: 23.5' },
        { x: 'Jul', y: 26, text: 'Jul: 26' }, { x: 'Aug', y: 25,
text: 'Aug: 25' },
        { x: 'Sep', y: 21, text: 'Sep: 21' }, { x: 'Oct', y: 15,
text: 'Oct: 15' } ],
      dataLabel: { visible: true, name: 'text', position: 'Outside'
},
      startAngle: 270,
      endAngle: 90,
      xName: 'x',
      yName: 'y'
    }
  ],
  '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>

```

```

</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Color & Text Mapping

The fill color and the text in the data source can be mapped to the chart using `pointColorMapping` in series and `name` in datalabel respectively.

INDEX.JS

```

var piechart = new ej.charts.AccumulationChart({
    series: [
        {
            dataSource: [
                { x: 'Jan', y: 3, text: 'Jan: 3' }, { x: 'Feb', y: 3.5,
text: 'Feb: 3.5' },
                { x: 'Mar', y: 7, text: 'Mar: 7' }, { x: 'Apr', y: 13.5,
text: 'Apr: 13.5' },
                { x: 'May', y: 19, text: 'May: 19' }, { x: 'Jun', y: 23.5,
text: 'Jun: 23.5' },
                { x: 'Jul', y: 26, text: 'Jul: 26' }, { x: 'Aug', y: 25,
text: 'Aug: 25' },
                { x: 'Sep', y: 21, text: 'Sep: 21' }, { x: 'Oct', y: 15,
text: 'Oct: 15' } ],
            dataLabel: { visible: true, name: 'text', position: 'Outside'
        },
        {
            startAngle: 270,
            endAngle: 90,
            xName: 'x',
            yName: 'y'
        }
    ]
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>

```

```

</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
</body>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customization

Individual points can be customized using the `pointRender` event.

INDEX.JS

```

var piechart = new ej.charts.AccumulationChart({
    series: [
        {
            dataSource: [
                { x: 'Jan', y: 3, text: 'Jan: 3' }, { x: 'Feb', y: 3.5,
text: 'Feb: 3.5' },
                { x: 'Mar', y: 7, text: 'Mar: 7' }, { x: 'Apr', y: 13.5,
text: 'Apr: 13.5' },
                { x: 'May', y: 19, text: 'May: 19' }, { x: 'Jun', y: 23.5,
text: 'Jun: 23.5' },
                { x: 'Jul', y: 26, text: 'Jul: 26' }, { x: 'Aug', y: 25,
text: 'Aug: 25' },
                { x: 'Sep', y: 21, text: 'Sep: 21' }, { x: 'Oct', y: 15,
text: 'Oct: 15' },
                { x: 'Nov', y: 9, text: 'Nov: 9' }, { x: 'Dec', y: 3.5,
text: 'Dec: 3.5' }],
            dataLabel: { visible: true, name: 'text', position: 'Outside' },
            groupTo: '11',
            xName: 'x',
            yName: 'y'
        }
    ],
    textRender: (args) => {
        if (args.text.indexOf('Others') > -1) {
            args.color = 'red';
            args.border.width = 1;
        }
    },
    pointRender: (args) => {
        if ((args.point.x).indexOf('Others') > -1) {
            args.fill = '#D3D3D3';
        }
    },
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Hide pie or doughnut border

By default, the border will appear in the pie/doughnut chart while mouse hover on the chart. You can disable the the border by setting `enableBorderOnMouseMove` property is `false`.

INDEX.JS

```

var dataMapping = [{ x: 'Jan', y: 3, fill: '#498fff', text: 'January' }, { x:
'Feb', y: 3.5, fill: '#ffa060', text: 'February' },
  { x: 'Mar', y: 7, fill: '#ff68b6', text: 'March' }, { x: 'Apr', y: 13.5,
fill: '#81e2a1', text: 'April' }];
var chart = new ej.charts.AccumulationChart({
  enableBorderOnMouseMove: false,
  series: [
    {
      dataSource: dataMapping,
      xName: 'x',
      yName: 'y',
    }
  ]
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">

```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Multi-level pie chart

You can achieve a multi-level drill down in pie and doughnut charts using [pointClick](#) event. If user clicks any point in the chart, that corresponding data will be shown in the next level and so on based on point clicked.

You can also achieve drill-up (back to the initial state) by using [chartMouseClicked](#) event. In below sample, you can drill-up by clicking back button in the center of the chart.

INDEX.JS

```

window.pointIndex = -1;
var data = [
    {
        x: 'SUV',
        y: 25,
        z: [
            {
                title: 'Automobile Sales in the SUV Segment',
                x: 'Toyota',
                y: 8,
                z: [
                    { x: '2000', y: 20 },
                    { x: '2001', y: 30 },
                    { x: '2002', y: 40 },
                ],
            },
            { x: 'Ford', y: 12 },
            { x: 'GM', y: 17 },
            { x: 'Renault', y: 6 },
        ],
    },
],

```



```

{
  x: 'Car',
  y: 37,
  z: [
    { title: 'Automobile Sales in the Car Segment', x: 'Toyota', y: 7 },
    { x: 'Chrysler', y: 12 },
    { x: 'Nissan', y: 9 },
    { x: 'Ford', y: 15 },
  ],
},
{
  x: 'Pickup',
  y: 15,
  z: [
    { title: 'Automobile Sales in the Pickup Segment', x: 'Nissan', y: 9
  },
    { x: 'Chrysler', y: 4 },
    { x: 'Ford', y: 7 },
    { x: 'Toyota', y: 20 },
  ],
},
{
  x: 'Minivan',
  y: 23,
  z: [
    { title: 'Automobile Sales in the Minivan Segment', x: 'Hummer', y: 11
  },
    { x: 'Ford', y: 5 },
    { x: 'GM', y: 12 },
    { x: 'Chrysler', y: 3 },
  ],
},
];
var click = function (args) {
  if (pie.series[0].name !== 'Level 3') {
    switch (args.pointIndex) {
      case 0:
        pie.series[0].dataSource = data[0].z[0].z;
        pie.title = 'SUV Sales by Years';
        pie.series[0].name = 'Level 3';
        grid.columns[0].headerText = 'Year';
        grid.refresh();
        pie.refresh();
        break;
    }
    grid.dataSource = pie.series[0].dataSource;
  }
};
var pointClick = function (args) {
  if (ej.charts.getElement(pie.element.id + '_Series_' + args.seriesIndex +
'_Point_' + args.pointIndex)) {
    pie.series[0].dataSource = data[args.pointIndex].z;
    pie.title = data[args.pointIndex].z[0].title;
    window.pointIndex = args.pointIndex;
    pie.series[0].name = 'Level 2';
    pie.series[0].innerRadius = '30%';
    pie.annotations = [

```

```

        {
            content:
                '<div id="back" style="cursor:pointer;padding:3px;width:30px;
height:30px;">' +
                '',
            region: 'Series',
            x: '50%',
            y: '50%',
        },
    ];
    pie.pointClick = click;
    var secondlevelpie = pie;
}
grid.dataSource = pie.series[0].dataSource;
grid.columns[0].headerText = data[args.pointIndex].x;
};
var instance = {
    series: [
        {
            dataSource: data,
            dataLabel: {
                visible: true,
                position: 'Inside',
            },
            radius: '70%',
            xName: 'x',
            yName: 'y',
            startAngle: 0,
            endAngle: 360,
            innerRadius: '0%',
            name: 'Level 1',
        },
    ],
    enableSmartLabels: false,
    legendSettings: { visible: true },
    chartMouseClick: (args) => {
        if (args.target.indexOf('back') > -1) {
            if (pie.series[0].name === 'Level 3') {
                pie.series[0].dataSource = data[window.pointIndex].z;
                pie.series[0].name = 'Level 2';
                pie.title = data[window.pointIndex].z[0].title;
                pie.series[0].innerRadius = '30%';
                grid.dataSource = pie.series[0].dataSource;
                grid.columns[0].headerText = data[window.pointIndex].x;
                grid.refresh();
                pie.refresh();
            } else if (pie.series[0].name === 'Level 2') {
                pie.series[0].dataSource = data;
                pie.series[0].name = 'Level 1';
                pie.series[0].innerRadius = '0%';
                pie.title = 'Automobile Sales by Category';
                pie.annotations = [{}];
                pie.pointClick = pointClick;
                grid.dataSource = pie.series[0].dataSource;
                grid.columns[0].headerText = 'Vehicle';
            }
        }
    }
};

```

```

    }
    }
    grid.dataSource = pie.series[0].dataSource;
  },
  pointClick: pointClick,
  title: 'Automobile Sales by Category',
};
var pie = new ej.charts.AccumulationChart(instance);
pie.appendTo('#container');
var grid = new ej.grids.Grid({
  dataSource: data,
  columns: [
    { field: 'x', headerText: 'Vehicle', type: 'string' },
    { field: 'y', headerText: 'Sales', type: 'string' },
  ],
});
grid.appendTo('#Grid');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet">
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div class="control-section">
    <div id="link">
      <a id="category" style="visibility:hidden; display:inline-block">
        Sales by Category
      </a>
      <p style="visibility:hidden; display:inline-block"
id="symbol">&#160;&#62;&#62;&#160;</p>
      <p id="text" style="display:inline-block;"></p>
    </div>
    <div id="container"></div>
  </div>

  <script>
var ele = document.getElementById('container');
```

```

if(ele) {
    ele.style.visibility = "visible";
}

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Data label](#)
- [Grouping](#)

Pyramid in ##Platform_Name## Accumulation chart control

To render a pyramid series, use the series [type](#) as `Pyramid` and inject `PyramidSeries` module using the `AccumulationChart.Inject(PyramidSeries)` method.

INDEX.JS

```

var chart = new ej.charts.AccumulationChart({
    width: '600px',
    series: [{
        type: 'Pyramid',
        dataSource: [{ x: 'Australia', y: 20, text: 'Australia 20%' },
            { x: 'France', y: 22, text: 'France 22%' },
            { x: 'China', y: 23, text: 'China 23%' },
            { x: 'India', y: 24, text: 'India 24%' },
            { x: 'Japan', y: 25, text: 'Japan 25%' },
            { x: 'Germany', y: 27, text: 'Germany 27%' } ],
        xName: 'x', yName: 'y',
        dataLabel: { name: 'text', visible: true, position: 'Inside' },
    }],
    title: 'Sales Distribution of Car by Region',
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>

```

```

</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Mode

The Pyramid chart supports linear and surface modes of rendering. The default type of the `pyramidMode` is `linear`.

INDEX.JS

```

var pyramidchart = new ej.charts.AccumulationChart({
    series: [{
        width: '600px',
        type: 'Pyramid',
        dataSource: [
            { x: 'Australia', y: 20, text: 'Australia 20%' },
            { x: 'France', y: 22, text: 'France 22%' },
            { x: 'China', y: 23, text: 'China 23%' },
            { x: 'India', y: 24, text: 'India 24%' },
            { x: 'Japan', y: 25, text: 'Japan 25%' },
            { x: 'Germany', y: 27, text: 'Germany 27%' },
        ],
        xName: 'x', yName: 'y',
        dataLabel: { visible: true, name: 'text', position: 'Inside' },
        pyramidMode: 'Linear',
    }],
    title: 'Sales Distribution of Car by Region',
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>

```

```

    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Size

The size of the pyramid chart can be customized by using the `width` and `height` properties.

INDEX.JS

```

var chart = new ej.charts.AccumulationChart({
    series: [{
        type: 'Funnel',
        dataSource: [{ x: 'Renewed', y: 18.20, text: 'Renewed 18.20%' },
            { x: 'Subscribe', y: 27.3, text: 'Subscribe 27.3%' },
            { x: 'Support', y: 55.9, text: 'Support 55.9%' },
            { x: 'Downloaded', y: 76.8, text: 'Downloaded 76.8%' },
            { x: 'Visited', y: 100, text: 'Visited 100%' }],
        xName: 'x', yName: 'y',
        //Width of the funnel will be 100% of the chart area
        width: '60%',
        //Height of the funnel will be 100% of the chart area
        height: '80%',
        dataLabel: { name: 'text', visible: true, position: 'Inside' },
    }],
    title: 'Website visitor',
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
</script>

```

```

var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Gap Between the Segments

Pyramid chart provides options to customize the space between the segments by using the `gapRatio` property of the series. It ranges from 0 to 1.

INDEX.JS

```

var pyramidchart = new ej.charts.AccumulationChart({
    series: [{
        type: 'Funnel',
        dataSource: [{ x: 'Renewed', y: 18.20, text: 'Renewed 18.20%' },
            { x: 'Subscribe', y: 27.3, text: 'Subscribe 27.3%' },
            { x: 'Support', y: 55.9, text: 'Support 55.9%' },
            { x: 'Downloaded', y: 76.8, text: 'Downloaded 76.8%' },
            { x: 'Visited', y: 100, text: 'Visited 100%' }],
        xName: 'x', yName: 'y',
        dataLabel: { visible: true, name: 'text', position: 'Inside' },
        // Defines the gap to be left between the segments
        gapRatio: 0.04,
    }],
    title: 'Website visitor',
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";

```

```

}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Explode

Points can be exploded on mouse click by setting the `explode` property to true. You can also explode the point on load using `explodeIndex`. Explode distance can be set by using `explodeOffset` property.

INDEX.JS

```

var piechart = new ej.charts.AccumulationChart({
  series: [
    {
      dataSource: [
        { x: 'Jan', y: 3, text: 'Jan: 3' }, { x: 'Feb', y: 3.5,
text: 'Feb: 3.5' },
        { x: 'Mar', y: 7, text: 'Mar: 7' }, { x: 'Apr', y: 13.5,
text: 'Apr: 13.5' },
        { x: 'May', y: 19, text: 'May: 19' }, { x: 'Jun', y: 23.5,
text: 'Jun: 23.5' },
        { x: 'Jul', y: 26, text: 'Jul: 26' }, { x: 'Aug', y: 25,
text: 'Aug: 25' },
        { x: 'Sep', y: 21, text: 'Sep: 21' }, { x: 'Oct', y: 15,
text: 'Oct: 15' },
        { x: 'Nov', y: 9, text: 'Nov: 9' }, { x: 'Dec', y: 3.5,
text: 'Dec: 3.5' }],
      dataLabel: { visible: true, name: 'text', position: 'Outside' },
      groupTo: '11',
      xName: 'x',
      yName: 'y',
      explode: true,
      //Specifies the distance of the point from the center during
explode, which takes values in both pixels and percentage
      explodeOffset: '10',
      //If set true, all the points in the series will get exploded on
load
      explodeAll: false,
      //Specifies index of the point, to be exploded on load
      explodeIndex: 2
    }
  ],
  textRender: (args) => {
    if (args.text.indexOf('Others') > -1) {
      args.color = 'red';
      args.border.width = 1;
    }
  },
  pointRender: (args) => {
    if ((args.point.x).indexOf('Others') > -1) {
      args.fill = '#D3D3D3';
    }
  },
}, '#element');

```


INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customization

Individual points can be customized using the `pointRender` event.

INDEX.JS

```

var piechart = new ej.charts.AccumulationChart({
  series: [
    {
      dataSource: [
        { x: 'Jan', y: 3, text: 'Jan: 3' }, { x: 'Feb', y: 3.5,
text: 'Feb: 3.5' },
        { x: 'Mar', y: 7, text: 'Mar: 7' }, { x: 'Apr', y: 13.5,
text: 'Apr: 13.5' },
        { x: 'May', y: 19, text: 'May: 19' }, { x: 'Jun', y: 23.5,
text: 'Jun: 23.5' },
        { x: 'Jul', y: 26, text: 'Jul: 26' }, { x: 'Aug', y: 25,
text: 'Aug: 25' },
        { x: 'Sep', y: 21, text: 'Sep: 21' }, { x: 'Oct', y: 15,
text: 'Oct: 15' },
        { x: 'Nov', y: 9, text: 'Nov: 9' }, { x: 'Dec', y: 3.5,
text: 'Dec: 3.5' }],
      dataLabel: { visible: true, name: 'text', position: 'Outside' },
      groupTo: '11',
      xName: 'x',
      yName: 'y'
    }
  ]
});

```

```

    ],
    textRender: (args) => {
        if (args.text.indexOf('Others') > -1) {
            args.color = 'red';
            args.border.width = 1;
        }
    },
    pointRender: (args) => {
        if ((args.point.x).indexOf('Others') > -1) {
            args.fill = '#D3D3D3';
        }
    },
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Data label](#)
- [Grouping](#)

Funnel in ##Platform_Name## Accumulation chart control

To render a funnel series, use the series [type](#) as `Funnel` and inject, the `FunnelSeries` module using the `AccumulationChart.Inject(FunnelSeries)` method.

INDEX.JS

```

var chart = new ej.charts.AccumulationChart({
  width: '500px',
  series: [{
    type: 'Funnel',
    dataSource: [{ x: 'Renewed', y: 18.20, text: 'Renewed 18.20%' },
      { x: 'Subscribe', y: 27.3, text: 'Subscribe 27.3%' },
      { x: 'Support', y: 55.9, text: 'Support 55.9%' },
      { x: 'Downloaded', y: 76.8, text: 'Downloaded 76.8%' },
      { x: 'Visited', y: 100, text: 'Visited 100%' }],
    xName: 'x', yName: 'y',
    dataLabel: { name: 'text', visible: true, position: 'Inside' },
    width: '60%', height: '90%'
  }],
  title: 'Website visitor',
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Size

The size of the funnel chart can be customized by using the `width` and `height` properties.

INDEX.JS

```

var chart = new ej.charts.AccumulationChart({
  series: [{
    type: 'Funnel',
    dataSource: [{ x: 'Renewed', y: 18.20, text: 'Renewed 18.20%' },

```

```

    { x: 'Subscribe', y: 27.3, text: 'Subscribe 27.3%' },
    { x: 'Support', y: 55.9, text: 'Support 55.9%' },
    { x: 'Downloaded', y: 76.8, text: 'Downloaded 76.8%' },
    { x: 'Visited', y: 100, text: 'Visited 100%' }],
    xName: 'x', yName: 'y',
    //Width of the funnel will be 100% of the chart area
    width: '60%',
    //Height of the funnel will be 100% of the chart area
    height: '80%',
    dataLabel: { name: 'text', visible: true, position: 'Inside' },
  ]],
  title: 'Website visitor',
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Neck Size

The funnel's neck size can be customized by using the `neckWidth` and `neckHeight` properties.

INDEX.JS

```

var chart = new ej.charts.AccumulationChart({
  series: [{
    type: 'Funnel',
    dataSource: [{ x: 'Renewed', y: 18.20, text: 'Renewed 18.20%' },
    { x: 'Subscribe', y: 27.3, text: 'Subscribe 27.3%' },
    { x: 'Support', y: 55.9, text: 'Support 55.9%' },

```

```

    { x: 'Downloaded', y: 76.8, text: 'Downloaded 76.8%' },
    { x: 'Visited', y: 100, text: 'Visited 100%' }]],
    xName: 'x', yName: 'y',
    //Width of the neck will be set as 25% of the chart area
    neckWidth: '25%', neckHeight: '5%',
    dataLabel: { name: 'text', visible: true, position: 'Inside' },
  ]],
  title: 'Website visitor',
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Gap Between the Segments

Funnel chart provides options to customize the space between the segments by using the `gapRatio` property of the series. It ranges from 0 to 1.

INDEX.JS

```

var pyramidchart = new ej.charts.AccumulationChart({
  series: [{
    type: 'Funnel',
    dataSource: [{ x: 'Renewed', y: 18.20, text: 'Renewed 18.20%' },
    { x: 'Subscribe', y: 27.3, text: 'Subscribe 27.3%' },
    { x: 'Support', y: 55.9, text: 'Support 55.9%' },
    { x: 'Downloaded', y: 76.8, text: 'Downloaded 76.8%' },
    { x: 'Visited', y: 100, text: 'Visited 100%' }]],
    xName: 'x', yName: 'y',

```

```

        dataLabel: { visible: true, name: 'text', position: 'Inside' },
        // Defines the gap to be left between the segments
        gapRatio: 0.04,
    }],
    title: 'Website visitor',
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Explode

Points can be exploded on mouse click by setting the `explode` property to true. You can also explode the point on load using `explodeIndex`. Explode distance can be set by using `explodeOffset` property.

INDEX.JS

```

var piechart = new ej.charts.AccumulationChart({
  series: [
    {
      dataSource: [
        { x: 'Jan', y: 3, text: 'Jan: 3' }, { x: 'Feb', y: 3.5,
text: 'Feb: 3.5' },
        { x: 'Mar', y: 7, text: 'Mar: 7' }, { x: 'Apr', y: 13.5,
text: 'Apr: 13.5' },
        { x: 'May', y: 19, text: 'May: 19' }, { x: 'Jun', y: 23.5,
text: 'Jun: 23.5' },
        { x: 'Jul', y: 26, text: 'Jul: 26' }, { x: 'Aug', y: 25,
text: 'Aug: 25' },

```

```

        { x: 'Sep', y: 21, text: 'Sep: 21' }, { x: 'Oct', y: 15,
text: 'Oct: 15' },
        { x: 'Nov', y: 9, text: 'Nov: 9' }, { x: 'Dec', y: 3.5,
text: 'Dec: 3.5' }]],
        dataLabel: { visible: true, name: 'text', position: 'Outside'
    },
    groupTo: '11',
    xName: 'x',
    yName: 'y'
    },
    ],
    textRender: (args: IAccTextRenderEventArgs) => {
        if (args.text.indexOf('Others') > -1) {
            args.color = 'red';
            args.border.width = 1;
        }
    },
    pointRender: (args: IAccPointRenderEventArgs) => {
        if ((args.point.x as string).indexOf('Others') > -1) {
            args.fill = '#D3D3D3';
        }
    },
    },
    '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

Smart Data Label

It provides the data label smart arrangement of the funnel and pyramid series. The overlap data label will be placed on left side of the funnel/pyramid series.

INDEX.JS

```

var data = [
  { 'x': 'China', y: 1409517397, text: 'China' },
  { 'x': 'India', y: 1339180127, text: 'India' },
  { 'x': 'United States', y: 324459463, text: 'United States' },
  { 'x': 'Indonesia', y: 263991379, text: 'Indonesia' },
  { 'x': 'Brazil', y: 209288278, text: 'Brazil' },
  { 'x': 'Pakistan', y: 197015955, text: 'Pakistan' },
  { 'x': 'Nigeria', y: 190886311, text: 'Nigeria' },
  { 'x': 'Bangladesh', y: 164669751, text: 'Bangladesh' },
  { 'x': 'Russia', y: 143989754, text: 'Russia' },
  { 'x': 'Mexico', y: 129163276, text: 'Mexico' },
  { 'x': 'Japan', y: 127484450, text: 'Japan' },
  { 'x': 'Ethiopia', y: 104957438, text: 'Ethiopia' },
  { 'x': 'Philippines', y: 104918090, text: 'Philippines' },
  { 'x': 'Egypt', y: 97553151, text: 'Egypt' },
  { 'x': 'Vietnam', y: 95540800, text: 'Vietnam' },
  { 'x': 'Germany', y: 82114224, text: 'Germany' }];
var chart = new ej.charts.AccumulationChart({
  series: [{
    type: 'Funnel', dataSource: data, xName: 'x', yName: 'y',
    neckWidth: '15%',
    neckHeight: '18%',
    name: '2017 Population',
    dataLabel: {
      visible: true, position: 'Outside',
      connectorStyle: { length: '6%' }, name: 'text',
    },
    explode: false,
  }],
  legendSettings: { visible: false },
  //Initializing tooltip
  tooltip: { enable: true, format: '${point.x} : <b>${point.y}</b>' },
  load: function (args) {
    var selectedTheme = location.hash.split('/')[1];
    selectedTheme = selectedTheme ? selectedTheme : 'Material';
    args.accumulation.theme = (selectedTheme.charAt(0).toUpperCase() +
      selectedTheme.slice(1)).replace(/-dark/i, 'Dark');
    if (args.accumulation.availableSize) {
      if (args.accumulation.availableSize.width <
args.accumulation.availableSize.height) {
        args.accumulation.series[0].width = '80%';
        args.accumulation.series[0].height = '70%';
      }
    }
  },
  resized: function (args) {
    var bounds =
document.getElementById('container').getBoundingClientRect();
    if (bounds.width < bounds.height) {
      args.accumulation.series[0].width = '80%';
      args.accumulation.series[0].height = '70%';
    }
  }
});

```



```

    } else {
        args.accumulation.series[0].width = '60%';
        args.accumulation.series[0].height = '80%';
    }
},
//Initializing Chart title
title: 'Top population countries in the world 2017',
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Customization

Individual points can be customized using the `pointRender` event.

INDEX.JS

```

var piechart = new ej.charts.AccumulationChart({
  series: [
    {
      dataSource: [
        { x: 'Renewed', y: 18.20, text: 'Renewed 18.20%' }, { x:
'Subscribe', y: 27.3, text: 'Subscribe 27.3%' },
        { x: 'Support', y: 55.9, text: 'Support 55.9%' }, { x:
'Downloaded', y: 76.8, text: 'Downloaded 76.8%' },
        { x: 'Visited', y: 100, text: 'Visited 100%' }
      ],
      type: 'Funnel',

```

```

        gapRatio: 0.08,
        xName: 'x',
        yName: 'y'
    }
],
pointRender: (args) => {
    if ((args.point.x).indexOf('Downloaded') > -1) {
        args.fill = '#f4bc42';
    }
    else {
        args.fill = '#597cf9';
    }
},
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Data label](#)
- [Grouping](#)

Data label in ##Platform_Name## Accumulation chart control

Data label can be added to a chart series by enabling the [visible](#)

option in the dataLabel property.

INDEX.JS

```

var piechart = new ej.charts.AccumulationChart({
  enableSmartLabels: true,
  series: [
    {
      dataSource: [
        { x: 'Jan', y: 3, text: 'Jan: 3' }, { x: 'Feb', y: 3.5,
text: 'Feb: 3.5' },
        { x: 'Mar', y: 7, text: 'Mar: 7' }, { x: 'Apr', y: 13.5,
text: 'Apr: 13.5' },
        { x: 'May', y: 19, text: 'May: 19' }, { x: 'Jun', y: 23.5,
text: 'Jun: 23.5' },
        { x: 'Jul', y: 26, text: 'Jul: 26' }, { x: 'Aug', y: 25,
text: 'Aug: 25' },
        { x: 'Sep', y: 21, text: 'Sep: 21' }, { x: 'Oct', y: 15,
text: 'Oct: 15' } ],
      dataLabel: { visible: true, name: 'text', position: 'Outside',
template: '<div>${point.x}</div><div>${point.y}</div>' },
      xName: 'x',
      yName: 'y'
    }
  ]
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Note: To use the data label feature, inject the `DataLabel` using the `Chart.Inject(DataLabel)` method.

Positioning

Accumulation chart provides support for placing the data label either `inside` or `outside` the chart.

INDEX.JS

```
var piechart = new ej.charts.AccumulationChart({
  enableSmartLabels: true,
  series: [
    {
      dataSource: [
        { x: 'Jan', y: 3, text: 'Jan: 3' }, { x: 'Feb', y: 3.5,
text: 'Feb: 3.5' },
        { x: 'Mar', y: 7, text: 'Mar: 7' }, { x: 'Apr', y: 13.5,
text: 'Apr: 13.5' },
        { x: 'May', y: 19, text: 'May: 19' }, { x: 'Jun', y: 23.5,
text: 'Jun: 23.5' },
        { x: 'Jul', y: 26, text: 'Jul: 26' }, { x: 'Aug', y: 25,
text: 'Aug: 25' },
        { x: 'Sep', y: 21, text: 'Sep: 21' }, { x: 'Oct', y: 15,
text: 'Oct: 15' } ],
      dataLabel: { visible: true, name: 'text', position: 'Outside',
template: '<div>${point.x}</div><div>${point.y}</div>' },
      xName: 'x',
      yName: 'y'
    }
  ],
  '#element');

```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}

```

```

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Smart labels

Data labels will be arranged smartly without overlapping with each other. You can enable or disable this feature using the [enableSmartLabels](#) property.

INDEX.JS

```

var piechart = new ej.charts.AccumulationChart({
  enableSmartLabels: true,
  series: [
    {
      dataSource: [
        { x: 'Jan', y: 3, text: 'Jan: 3' }, { x: 'Feb', y: 3.5,
text: 'Feb: 3.5' },
        { x: 'Mar', y: 7, text: 'Mar: 7' }, { x: 'Apr', y: 13.5,
text: 'Apr: 13.5' },
        { x: 'May', y: 19, text: 'May: 19' }, { x: 'Jun', y: 23.5,
text: 'Jun: 23.5' },
        { x: 'Jul', y: 26, text: 'Jul: 26' }, { x: 'Aug', y: 25,
text: 'Aug: 25' },
        { x: 'Sep', y: 21, text: 'Sep: 21' }, { x: 'Oct', y: 15,
text: 'Oct: 15' } ],
      dataLabel: { visible: true, name: 'text', position: 'Outside' },
      xName: 'x',
      yName: 'y'
    }
  ]
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');

```

```

if(ele) {
    ele.style.visibility = "visible";
}

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Data Label Template

Label content can be formatted by using the template option. Inside the template, you can add the placeholder text `${point.x}` and `${point.y}` to display corresponding data points x & y value. Using [template](#) property, you can set data label template in chart.

INDEX.JS

```

var piechart = new ej.charts.AccumulationChart({
    enableSmartLabels: true,
    series: [
        {
            dataSource: [
                { x: 'Jan', y: 3, text: 'Jan: 3' }, { x: 'Feb', y: 3.5,
text: 'Feb: 3.5' },
                { x: 'Mar', y: 7, text: 'Mar: 7' }, { x: 'Apr', y: 13.5,
text: 'Apr: 13.5' },
                { x: 'May', y: 19, text: 'May: 19' }, { x: 'Jun', y: 23.5,
text: 'Jun: 23.5' },
                { x: 'Jul', y: 26, text: 'Jul: 26' }, { x: 'Aug', y: 25,
text: 'Aug: 25' },
                { x: 'Sep', y: 21, text: 'Sep: 21' }, { x: 'Oct', y: 15,
text: 'Oct: 15' } ],
            dataLabel: { visible: true, name: 'text', position: 'Outside',
template: '<div>${point.x}</div><div>${point.y}</div>' },
            xName: 'x',
            yName: 'y'
        }
    ]
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```

```

<div id="container">
  <div id="element"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Connector Line

Connector line will be visible when the data label is placed outside the chart. The connector line can be customized using the `type`, `color`, `width`, `length` and `dashArray` properties

INDEX.JS

```

var piechart = new ej.charts.AccumulationChart({
  enableSmartLabels: true,
  series: [
    {
      dataSource: [
        { x: 'Jan', y: 13, text: 'Jan: 13' },
        { x: 'Feb', y: 13, text: 'Feb: 13' },
        { x: 'Mar', y: 17, text: 'Mar: 17' },
        { x: 'Apr', y: 13.5, text: 'Apr: 13.5' }],
      dataLabel: {
        visible: true, name: 'text', position: 'Outside',
        connectorStyle: {
          //Length of the connector line in pixels
          length: '50px',
          //Width of the connector line in pixels
          width: 2,
          //dashArray of the connector line
          dashArray: '5,3',
          //Color of the connector line
          color: '#f4429e',
          //Specifies the type of the connector line either Line
          or Curve
          type: 'Curve'
        }
      },
      xName: 'x',
      yName: 'y'
    }
  ],
  '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">

```

```

<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Text Mapping

Text from the data source can be mapped to data label using `name` property.

INDEX.JS

```

var piechart = new ej.charts.AccumulationChart({
    series: [
        {
            dataSource: [
                { x: 'Jan', y: 3, text: 'Jan: 3' }, { x: 'Feb', y: 3.5,
text: 'Feb: 3.5' },
                { x: 'Mar', y: 7, text: 'Mar: 7' }, { x: 'Apr', y: 13.5,
text: 'Apr: 13.5' },
                { x: 'May', y: 19, text: 'May: 19' }, { x: 'Jun', y: 23.5,
text: 'Jun: 23.5' },
                { x: 'Jul', y: 26, text: 'Jul: 26' }, { x: 'Aug', y: 25,
text: 'Aug: 25' },
                { x: 'Sep', y: 21, text: 'Sep: 21' }, { x: 'Oct', y: 15,
text: 'Oct: 15' } ],
            dataLabel: { visible: true, name: 'text', position: 'Outside'
},
            startAngle: 270,
            endAngle: 90,
            xName: 'x',
            yName: 'y'
        }
    ]
}, '#element');

```

INDEX.HTML


```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Format

Data label for the accumulation chart can be formatted using [format](#) property. You can use the global formatting options, such as 'n', 'p', and 'c'.

INDEX.JS

```

var piechart = new ej.charts.AccumulationChart({
  series: [
    {
      dataSource: [
        { x: 'Jan', y: 3, text: 'Jan: 3' }, { x: 'Feb', y: 3, text:
'Feb: 3.5' },
        { x: 'Mar', y: 7, text: 'Mar: 7' }, { x: 'Apr', y: 13, text:
'Apr: 13.5' },
        { x: 'May', y: 19, text: 'May: 19' }, { x: 'Jun', y: 23,
text: 'Jun: 23.5' },
        { x: 'Jul', y: 26, text: 'Jul: 26' }, { x: 'Aug', y: 25,
text: 'Aug: 25' },
        { x: 'Sep', y: 21, text: 'Sep: 21' }, { x: 'Oct', y: 15,
text: 'Oct: 15' } ],
      dataLabel: { visible: true, position: 'Outside' , format: 'n2'
    },
    {
      startAngle: 270,
      endAngle: 90,
      xName: 'x',
      yName: 'y'
    }
  ]
}

```

```
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Value	Format	Resultant Value	Description
1000	n1	1000.0	The number is rounded to 1 decimal place.
1000	n2	1000.00	The number is rounded to 2 decimal places.
1000	n3	1000.000	The number is rounded to 3 decimal place.
0.01	p1	1.0%	The number is converted to percentage with 1 decimal place.
0.01	p2	1.00%	The number is converted to percentage with 2 decimal place.
0.01	p3	1.000%	The number is converted to percentage with 3 decimal place.
1000	c1	\$1000.0	The currency symbol is appended to number and number is rounded to 1 decimal place.
1000	c2	\$1000.00	The currency symbol is appended to number and number is rounded to 2 decimal place.

Customization

Individual text can be customized using the `textRender` event.

INDEX.JS

```
var piechart = new ej.charts.AccumulationChart({
  series: [
    {
      dataSource: [
        { x: 'Jan', y: 3, text: 'Jan: 3' }, { x: 'Feb', y: 3.5,
text: 'Feb: 3.5' },
        { x: 'Mar', y: 7, text: 'Mar: 7' }, { x: 'Apr', y: 13.5,
text: 'Apr: 13.5' },
        { x: 'May', y: 19, text: 'May: 19' }, { x: 'Jun', y: 23.5,
text: 'Jun: 23.5' },
        { x: 'Jul', y: 26, text: 'Jul: 26' }, { x: 'Aug', y: 25,
text: 'Aug: 25' },
        { x: 'Sep', y: 21, text: 'Sep: 21' }, { x: 'Oct', y: 15,
text: 'Oct: 15' },
        { x: 'Nov', y: 9, text: 'Nov: 9' }, { x: 'Dec', y: 3.5,
text: 'Dec: 3.5' }],
      dataLabel: { visible: true, name: 'text', position: 'Outside' },
      groupTo: '11',
      xName: 'x',
      yName: 'y'
    }
  ],
  textRender: (args) => {
    if (args.text.indexOf('Others') > -1) {
      args.color = 'red';
      args.border.width = 1;
    }
  },
  pointRender: (args) => {
    if ((args.point.x).indexOf('Others') > -1) {
      args.fill = '#D3D3D3';
    }
  },
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
```

```
<body>

  <div id="container">
    <div id="element"></div>
  </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Text wrap

When the data label text exceeds the container, the text can be wrapped by using [textWrap](#) property. End user can also wrap the data label text based on [maxWidth](#) property.

INDEX.JS

```
var piechart = new ej.charts.AccumulationChart({
  enableSmartLabels: true,
  series: [
    {
      dataSource: [
        { x: 'Chrome', y: 100, text: 'Chrome (100M)<br>40%',
        tooltipMappingName: '40%' },
        { x: 'UC Browser', y: 40, text: 'UC Browser (40M)<br>16%',
        tooltipMappingName: '16%' },
        { x: 'Opera', y: 30, text: 'Opera (30M)<br>12%',
        tooltipMappingName: '12%' },
        { x: 'Safari', y: 30, text: 'Safari (30M)<br>12%',
        tooltipMappingName: '12%' },
        { x: 'Firefox', y: 25, text: 'Firefox (25M)<br>10%',
        tooltipMappingName: '10%' },
        { x: 'Others', y: 25, text: 'Others (25M)<br>10%',
        tooltipMappingName: '10%' }
      ],
      xName: 'x',
      yName: 'y',
      startAngle: 270,
      endAngle: 90,
      explode: true, radius: '100%',
      innerRadius: '40%', tooltipMappingName: 'tooltipMappingName',
      dataLabel: {
        visible: true, position: 'Inside'
      },
      maxWidth: 100, textWrap: 'Wrap',
      name: 'text', enableRotation: true,
    }
  ],
}, '#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Show percentages in data labels of pie chart

You can show the percentages in data labels of pie chart using `textRender` event and `template` option.

Using textRender event

You can customize the data label of pie chart using `textRender` event as follows to show percentage..

INDEX.JS

```

var piechart = new ej.charts.AccumulationChart({
  enableSmartLabels: true,
  series: [
    {
      dataSource: [
        { x: 'Jan', y: 3, text: 'Jan: 3' }, { x: 'Feb', y: 3.5,
text: 'Feb: 3.5' },
        { x: 'Mar', y: 7, text: 'Mar: 7' }, { x: 'Apr', y: 13.5,
text: 'Apr: 13.5' },
        { x: 'May', y: 19, text: 'May: 19' }, { x: 'Jun', y: 23.5,
text: 'Jun: 23.5' },
        { x: 'Jul', y: 26, text: 'Jul: 26' }, { x: 'Aug', y: 25,
text: 'Aug: 25' },
        { x: 'Sep', y: 21, text: 'Sep: 21' }, { x: 'Oct', y: 15,
text: 'Oct: 15' } ],
      dataLabel: { visible: true },
      xName: 'x',
      yName: 'y'
    }
  ],

```

```

    textRender: function(args) {
        args.text = args.point.percentage + "%";
    }
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Using template

You can display the percentage values in data label of pie chart using `template` option.

INDEX.JS

```

var piechart = new ej.charts.AccumulationChart({
    enableSmartLabels: true,
    series: [
        {
            dataSource: [
                { x: 'Jan', y: 3, text: 'Jan: 3' }, { x: 'Feb', y: 3.5,
text: 'Feb: 3.5' },
                { x: 'Mar', y: 7, text: 'Mar: 7' }, { x: 'Apr', y: 13.5,
text: 'Apr: 13.5' },
                { x: 'May', y: 19, text: 'May: 19' }, { x: 'Jun', y: 23.5,
text: 'Jun: 23.5' },
                { x: 'Jul', y: 26, text: 'Jul: 26' }, { x: 'Aug', y: 25,
text: 'Aug: 25' },
                { x: 'Sep', y: 21, text: 'Sep: 21' }, { x: 'Oct', y: 15,
text: 'Oct: 15' }],

```

```

        dataLabel: { visible: true, template: "<div
id='dataLabelTemplate'>${point.percentage}%</div>" },
        xName: 'x',
        yName: 'y'
    }
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

Grouping in ##Platform_Name## Accumulation chart control

You can club/group few points of the series based on

[groupTo](#) property. For example, if the club

value is 11, then the points with value less than 11 is grouped together and will be showed as a single point with label **others**. The property also takes value in percentage (percentage of total data points value).

INDEX.JS

```

var piechart = new ej.charts.AccumulationChart({
    series: [
        {
            dataSource: [
                { x: 'Jan', y: 3, text: 'Jan: 3' }, { x: 'Feb', y: 3.5,
text: 'Feb: 3.5' },

```

```

        { x: 'Mar', y: 7, text: 'Mar: 7' }, { x: 'Apr', y: 13.5,
text: 'Apr: 13.5' },
        { x: 'May', y: 19, text: 'May: 19' }, { x: 'Jun', y: 23.5,
text: 'Jun: 23.5' },
        { x: 'Jul', y: 26, text: 'Jul: 26' }, { x: 'Aug', y: 25,
text: 'Aug: 25' },
        { x: 'Sep', y: 21, text: 'Sep: 21' }, { x: 'Oct', y: 15,
text: 'Oct: 15' },
        { x: 'Nov', y: 9, text: 'Nov: 9' }, { x: 'Dec', y: 3.5,
text: 'Dec: 3.5' }],
        dataLabel: { visible: true, name: 'text', position: 'Outside' },
        groupTo: '11',
        xName: 'x',
        yName: 'y'
    }
},
textRender: (args) => {
    if (args.text.indexOf('Others') > -1) {
        args.color = 'red';
        args.border.width = 1;
    }
},
pointRender: (args) => {
    if ((args.point.x).indexOf('Others') > -1) {
        args.fill = '#D3D3D3';
    }
},
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>

```



```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Group Mode

Slice can also be grouped based on number of points by specifying the `groupMode` to `Point`. For example, if the group to value is 11, accumulation chart will show 1st 11 points and will group remaining entries in the collection as a single point.

INDEX.JS

```
var piechart = new ej.charts.AccumulationChart({
  series: [
    {
      dataSource: [
        { x: 'Jan', y: 3, text: 'Jan: 3' }, { x: 'Feb', y: 3.5,
text: 'Feb: 3.5' },
        { x: 'Mar', y: 7, text: 'Mar: 7' }, { x: 'Apr', y: 13.5,
text: 'Apr: 13.5' },
        { x: 'May', y: 19, text: 'May: 19' }, { x: 'Jun', y: 23.5,
text: 'Jun: 23.5' },
        { x: 'Jul', y: 26, text: 'Jul: 26' }, { x: 'Aug', y: 25,
text: 'Aug: 25' },
        { x: 'Sep', y: 21, text: 'Sep: 21' }, { x: 'Oct', y: 15,
text: 'Oct: 15' },
        { x: 'Nov', y: 9, text: 'Nov: 9' }, { x: 'Dec', y: 3.5,
text: 'Dec: 3.5' }],
      dataLabel: { visible: true, name: 'text', position: 'Outside' },
      groupTo: '4',
      groupMode: 'Point',
      xName: 'x',
      yName: 'y'
    }
  ],
  textRender: (args) => {
    if (args.text.indexOf('Others') > -1) {
      args.color = 'red';
      args.border.width = 1;
    }
  },
  pointRender: (args) => {
    if ((args.point.x).indexOf('Others') > -1) {
      args.fill = '#D3D3D3';
    }
  },
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customization

You can customize the grouped point and its data label using `pointRender` and `textRender` event.

INDEX.JS

```

var piechart = new ej.charts.AccumulationChart({
    series: [
        {
            dataSource: [
                { x: 'Jan', y: 3 }, { x: 'Feb', y: 3.5 },
                { x: 'Mar', y: 7 }, { x: 'Apr', y: 13.5 },
                { x: 'May', y: 19 }, { x: 'Jun', y: 23.5 },
                { x: 'Jul', y: 26 }, { x: 'Aug', y: 25 },
                { x: 'Sep', y: 21 }, { x: 'Oct', y: 15 },
                { x: 'Nov', y: 9 }, { x: 'Dec', y: 3.5 }],
            dataLabel: { visible: true, name: 'text', position: 'Outside' },
            groupTo: '11',
            xName: 'x',
            yName: 'y'
        }
    ],
    textRender: (args) => {
        if (args.text.indexOf('Others') > -1) {
            args.text = 'Grouped Slices';
            args.color = 'red';
            args.border.width = 1;
        }
    },
    pointRender: (args) => {
        if ((args.point.x).indexOf('Others') > -1) {
            args.fill = '#D3D3D3';
        }
    },
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Empty points in ##Platform_Name## Accumulation chart control

The data points those uses the **null** or **undefined** as value are considered as empty points. The empty data points are ignored and not plotted in the chart. You can customize those points, using the **emptyPointSettings** property in series. The default mode of the empty point is **Gap**. Other supported modes are **Average** and **Zero**.

INDEX.TS

```

import { AccumulationChart, AccumulationDataLabel } from '@syncfusion/ej2-
charts';
AccumulationChart.Inject(AccumulationDataLabel);
let piechart: AccumulationChart = new AccumulationChart({
  series: [
    {
      dataSource: [{ x: 'Jan', y: 3 }, { x: 'Feb', y: 3.5 }, { x:
'Mar', y: undefined }, { x: 'Apr', y: 13.5 },
      { x: 'May', y: 19 }, { x: 'Jun', y: 23.5 }, { x: 'Jul', y: null
}, { x: 'Aug', y: 25 },
      { x: 'Sep', y: 21 }, { x: 'Oct', y: 15 }],
      xName: 'x',
      yName: 'y',
      emptyPointSettings: { mode: 'Zero', fill: 'pink'},
      dataLabel: { visible: true, position: 'Outside' }
    }
  ]
}

```

```
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Customization

Specific color for an empty point can be set by using the `fill` property in `emptyPointSettings` and the border for an empty point can be set by using the `border` property.

INDEX.TS

```
import { AccumulationChart } from '@syncfusion/ej2-charts';
let piechart: AccumulationChart = new AccumulationChart({
  series: [
    {
      dataSource: [{ x: 'Jan', y: 3 }, { x: 'Feb', y: 3.5 }, { x:
'Mar', y: undefined }, { x: 'Apr', y: 13.5 },
      { x: 'May', y: 19 }, { x: 'Jun', y: 23.5 }, { x: 'Jul', y: null
}, { x: 'Aug', y: 25 },
      { x: 'Sep', y: 21 }, { x: 'Oct', y: 15 }],
      emptyPointSettings: { mode: 'Average', fill: 'pink' },
      xName: 'x',
      yName: 'y'
    }
  ]
}, '#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Annotation in ##Platform_Name## Accumulation chart control

The annotations are used to mark the specific area of interest in the chart area with texts, shapes or images.

<!-- markdownlint-disable MD033 -->

By using the `<code>content</code>` option of annotation property, you can specify the Id of the element that needs to be displayed in the chart area

INDEX.JS

```

var piechart = new ej.charts.AccumulationChart({
  series: [
    {
      dataSource: [
        { x: 'Jan', y: 3, text: 'Jan: 3' }, { x: 'Feb', y: 3.5,
text: 'Feb: 3.5' },
        { x: 'Mar', y: 7, text: 'Mar: 7' }, { x: 'Apr', y: 13.5,
text: 'Apr: 13.5' },
        { x: 'May', y: 19, text: 'May: 19' }, { x: 'Jun', y: 23.5,
text: 'Jun: 23.5' },
        { x: 'Jul', y: 26, text: 'Jul: 26' }, { x: 'Aug', y: 25,
text: 'Aug: 25' },
        { x: 'Sep', y: 21, text: 'Sep: 21' }, { x: 'Oct', y: 15,
text: 'Oct: 15' } ],
      xName: 'x',

```

```

        yName: 'y'
    }
],
annotations:[{
    content: 'Feb-3.5',
    region: 'Series',
    coordinateUnits: 'Point',
    x: 'Feb',
    y: 3.5
}],
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Note: To use the annotations feature, inject the **AccumulationAnnotation** using the **Chart.Inject(AccumulationAnnotation)** method.

Region

The annotation can be placed with respect to either **Series** or **Chart**.

INDEX.JS

```

var piechart = new ej.charts.AccumulationChart({
  series: [
    {
      dataSource: [

```

```

        { x: 'Jan', y: 3, text: 'Jan: 3' }, { x: 'Feb', y: 3.5,
text: 'Feb: 3.5' },
        { x: 'Mar', y: 7, text: 'Mar: 7' }, { x: 'Apr', y: 13.5,
text: 'Apr: 13.5' },
        { x: 'May', y: 19, text: 'May: 19' }, { x: 'Jun', y: 23.5,
text: 'Jun: 23.5' },
        { x: 'Jul', y: 26, text: 'Jul: 26' }, { x: 'Aug', y: 25,
text: 'Aug: 25' },
        { x: 'Sep', y: 21, text: 'Sep: 21' }, { x: 'Oct', y: 15,
text: 'Oct: 15' }],
        xName: 'x',
        yName: 'y'
    }
],
    annotations:[{
        content: 'Feb-3.5',
        region: 'Series',
        coordinateUnits: 'Point',
        x: 'Feb',
        y: 3.5
    }],
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

Co-ordinate Units

Specifies the coordinate units of an annotation either in **Pixel** or **Point**.

INDEX.JS

```

var piechart = new ej.charts.AccumulationChart({
  series: [
    {
      dataSource: [
        { x: 'Jan', y: 3, text: 'Jan: 3' }, { x: 'Feb', y: 3.5,
text: 'Feb: 3.5' },
        { x: 'Mar', y: 7, text: 'Mar: 7' }, { x: 'Apr', y: 13.5,
text: 'Apr: 13.5' },
        { x: 'May', y: 19, text: 'May: 19' }, { x: 'Jun', y: 23.5,
text: 'Jun: 23.5' },
        { x: 'Jul', y: 26, text: 'Jul: 26' }, { x: 'Aug', y: 25,
text: 'Aug: 25' },
        { x: 'Sep', y: 21, text: 'Sep: 21' }, { x: 'Oct', y: 15,
text: 'Oct: 15' } ],
      xName: 'x',
      yName: 'y'
    }
  ],
  annotations:[{
    content: 'Feb-3.5',
    region: 'Series',
    coordinateUnits: 'Point',
    x: 'Feb',
    y: 3.5
  }],
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>

```



```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Alignment

The annotations can be moved vertically and horizontally from its default position by using `verticalAlignment` or `horizontalAlignment` properties. The `verticalAlignment` property takes value as `Top`, `Bottom` or `Middle` and the `horizontalAlignment` property takes value as `Near`, `Far` or `Center`.

INDEX.JS

```
var piechart = new ej.charts.AccumulationChart({
  series: [
    {
      dataSource: [
        { x: 'Jan', y: 3, text: 'Jan: 3' }, { x: 'Feb', y: 3.5,
text: 'Feb: 3.5' },
        { x: 'Mar', y: 7, text: 'Mar: 7' }, { x: 'Apr', y: 13.5,
text: 'Apr: 13.5' },
        { x: 'May', y: 19, text: 'May: 19' }, { x: 'Jun', y: 23.5,
text: 'Jun: 23.5' },
        { x: 'Jul', y: 26, text: 'Jul: 26' }, { x: 'Aug', y: 25,
text: 'Aug: 25' },
        { x: 'Sep', y: 21, text: 'Sep: 21' }, { x: 'Oct', y: 15,
text: 'Oct: 15' } ],
      xName: 'x',
      yName: 'y'
    }
  ],
  annotations:[{
    content: 'Feb-3.5',
    region: 'Series',
    coordinateUnits: 'Point',
    x: 'Feb',
    y: 3.5
  }],
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
```

```

<div id="container">
  <div id="element"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Tooltip in ##Platform_Name## Accumulation chart control

Tooltip for the accumulation chart can be enabled by using the [enable](#) property.

INDEX.JS

```

var piechart = new ej.charts.AccumulationChart({
  series: [
    {
      dataSource: [{ x: 'Jan', y: 3 }, { x: 'Feb', y: 3.5 }, { x:
'Mar', y: 7 }, { x: 'Apr', y: 13.5 },
      { x: 'May', y: 19 }, { x: 'Jun', y: 23.5 }, { x: 'Jul', y: 26 },
{ x: 'Aug', y: 25 },
      { x: 'Sep', y: 21 }, { x: 'Oct', y: 15 }],
      radius: '100%',
      xName: 'x',
      yName: 'y'
    }
  ],
  tooltip:{enable: true}
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
<script>

```

```

var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Note: To use tooltip feature, inject the `AccumulationTooltip` using the `Chart.Inject(AccumulationTooltip)` method.

Header

We can specify header for the tooltip using [header](#) property.

INDEX.JS

```

var piechart = new ej.charts.AccumulationChart({
    series: [
        {
            dataSource: [{ x: 'Jan', y: 3 }, { x: 'Feb', y: 3.5 }, { x:
'Mar', y: 7 }, { x: 'Apr', y: 13.5 },
            { x: 'May', y: 19 }, { x: 'Jun', y: 23.5 }, { x: 'Jul', y: 26 },
{ x: 'Aug', y: 25 },
            { x: 'Sep', y: 21 }, { x: 'Oct', y: 15 }],
            radius: '100%',
            xName: 'x',
            yName: 'y'
        }
    ],
    tooltip: { enable: true, header: "Pie Chart" }
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');

```

```

if(ele) {
    ele.style.visibility = "visible";
}

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Format

By default, tooltip shows information of x and y value in points. In addition to that, you can show more information in tooltip. For example the format `${series.name} ${point.x}` shows series name and point x value.

INDEX.JS

```

var piechart = new ej.charts.AccumulationChart({
    series: [
        {
            dataSource: [{ x: 'Jan', y: 3 }, { x: 'Feb', y: 3.5 }, { x:
'Mar', y: 7 }, { x: 'Apr', y: 13.5 },
            { x: 'May', y: 19 }, { x: 'Jun', y: 23.5 }, { x: 'Jul', y: 26 },
{ x: 'Aug', y: 25 },
            { x: 'Sep', y: 21 }, { x: 'Oct', y: 15 }],
            radius: '100%',
            xName: 'x',
            yName: 'y'
        }
    ],
    tooltip: { enable: true, format: '${point.x} : <b>${point.y}%</b>' }
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";

```

```

}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Tooltip format

Any HTML element can be displayed in the tooltip by using the [template](#) property.

INDEX.JS

```

var piechart = new ej.charts.AccumulationChart({
  series: [
    {
      dataSource: [{ x: 'Jan', y: 3 }, { x: 'Feb', y: 3.5 }, { x:
'Mar', y: 7 }, { x: 'Apr', y: 13.5 },
      { x: 'May', y: 19 }, { x: 'Jun', y: 23.5 }, { x: 'Jul', y: 26 },
{ x: 'Aug', y: 25 },
      { x: 'Sep', y: 21 }, { x: 'Oct', y: 15 }],
      radius: '100%',
      xName: 'x',
      yName: 'y'
    }
  ],
  tooltip: {
    enable: true,
    template: "<div id='templateWrap' style='background-
color:#bd18f9;border-radius: 3px; float: right;padding: 2px;line-height:
20px;text-align: center;'>" +
      "<img src='sun_annotation.png' />" +
      "<div style='color:white; font-family:Roboto; font-style:
medium; font-size:14px;float: right;padding: 2px;line-height: 20px;text-
align: center;padding-right:6px'><span>${y}</span></div></div>"
  }
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>

```

```
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Fixed tooltip

By default, tooltip track the mouse movement, but you can set a fixed position for the tooltip by using the [location](#) property.

INDEX.JS

```
var piechart = new ej.charts.AccumulationChart({
    series: [
        {
            dataSource: [{ x: 'Jan', y: 3 }, { x: 'Feb', y: 3.5 }, { x:
'Mar', y: 7 }, { x: 'Apr', y: 13.5 },
            { x: 'May', y: 19 }, { x: 'Jun', y: 23.5 }, { x: 'Jul', y: 26 },
{ x: 'Aug', y: 25 },
            { x: 'Sep', y: 21 }, { x: 'Oct', y: 15 }],
            radius: '100%',
            xName: 'x',
            yName: 'y'
        }
    ],
    tooltip: {
        enable: true,
        location: { x: 200, y: 20 }
    }
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
```

```
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Customization

The [fill](#) and [border](#) properties are used to customize the background color and border of the tooltip respectively. The [textStyle](#) property in the tooltip is used to customize the font of the tooltip text. The [highlightColor](#) property can be used to change the color of the data point when hovering.

INDEX.JS

```
var piechart = new ej.charts.AccumulationChart({
    series: [
        {
            dataSource: [{ x: 'Jan', y: 3 }, { x: 'Feb', y: 3.5 }, { x:
'Mar', y: 7 }, { x: 'Apr', y: 13.5 },
            { x: 'May', y: 19 }, { x: 'Jun', y: 23.5 }, { x: 'Jul', y: 26 },
{ x: 'Aug', y: 25 },
            { x: 'Sep', y: 21 }, { x: 'Oct', y: 15 }],
            radius: '100%',
            xName: 'x',
            yName: 'y'
        }
    ],
    highlightColor: 'red',
    tooltip: {
        enable: true,
        format: '${series.name} ${point.x} : ${point.y}',
        //fill for tooltip
        fill: '#7bb4eb',
        //border for tooltip
        border: {
            width: 2,
            color: 'grey'
        }
    }
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

To customize individual tooltip

Using [tooltipRender](#) event, you can customize a tooltip for particular point. event, you can customize a tooltip for particular point.

INDEX.JS

```

var piechart = new ej.charts.AccumulationChart({
    series: [
        {
            dataSource: [{ x: 'Jan', y: 3 }, { x: 'Feb', y: 3.5 }, { x:
'Mar', y: 7 }, { x: 'Apr', y: 13.5 },
            { x: 'May', y: 19 }, { x: 'Jun', y: 23.5 }, { x: 'Jul', y: 26 },
{ x: 'Aug', y: 25 },
            { x: 'Sep', y: 21 }, { x: 'Oct', y: 15 }],
            radius: '100%',
            xName: 'x',
            yName: 'y'
        }
    ],
    tooltip: {
        enable: true
    },
    tooltipRender: function(args) {
        if (args.point.index === 3) {
            args.text = args.point.x + ' ' + ':' + args.point.y + ' ' + ' ' +
'customtext';
            args.textStyle.color = '#f48042';
        }
    }
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">

```



```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Legend in ##Platform_Name## Accumulation chart control

As like a chart, the legend is also available for accumulation charts, which gives information about the points. By default, the legend will be placed on the right, if the width of the chart is high or will be placed on the bottom, if the height of the chart is high. Other customization features regarding the legend element are same as the [chart legend](#). Here, the legend for a point can be collapsed by giving the empty string to the x value of the point.

INDEX.JS

```

var piechart = new ej.charts.AccumulationChart({
    series: [
        {
            dataSource: [
                { x: 'Jan', y: 3, text: 'Jan: 3' }, { x: 'Feb', y: 3.5,
text: 'Feb: 3.5' },
                { x: 'Mar', y: 7, text: 'Mar: 7' }, { x: 'Apr', y: 13.5,
text: 'Apr: 13.5' },
                { x: 'May', y: 19, text: 'May: 19' }, { x: 'Jun', y: 23.5,
text: 'Jun: 23.5' },
                { x: 'Jul', y: 26, text: 'Jul: 26' }, { x: 'Aug', y: 25,
text: 'Aug: 25' },
                { x: 'Sep', y: 21, text: 'Sep: 21' }, { x: 'Oct', y: 15,
text: 'Oct: 15' } ],
            xName: 'x',
            yName: 'y'
        }
    ],
    legendSettings: {
        position: 'Right',
        visible: true,

```

```

        height: '40',
        width: '160'
    }
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Note: To use the legends feature, inject the `AccumulationLegend` using the `Chart.Inject(AccumulationLegend)` method.

Position and Alignment

By using the position property, you can position the legend at the `left`, `right`, `top` or `bottom` of the chart. You can also align the legend to `center`, `far` or `near` of the chart using the alignment property.

INDEX.JS

```

var piechart = new ej.charts.AccumulationChart({
  series: [
    {
      dataSource: [
        { x: 'Jan', y: 3, text: 'Jan: 3' }, { x: 'Feb', y: 3.5,
text: 'Feb: 3.5' },
        { x: 'Mar', y: 7, text: 'Mar: 7' }, { x: 'Apr', y: 13.5,
text: 'Apr: 13.5' },
        { x: 'May', y: 19, text: 'May: 19' }, { x: 'Jun', y: 23.5,
text: 'Jun: 23.5' },

```

```

        { x: 'Jul', y: 26, text: 'Jul: 26' }, { x: 'Aug', y: 25,
text: 'Aug: 25' },
        { x: 'Sep', y: 21, text: 'Sep: 21' }, { x: 'Oct', y: 15,
text: 'Oct: 15' } ],
        xName: 'x',
        yName: 'y'
    }
],
legendSettings: {
    position: 'Right',
    visible: true,
    height: '40',
    width: '160'
}
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Legend Reverse

You can reverse the order of the legend items by using the [reverse](#) property. By default, legend for the first series in the collection will be placed first.

INDEX.JS

```

var piechart = new ej.charts.AccumulationChart({
  series: [
    {

```

```

        dataSource: [
            { x: 'Jan', y: 3, text: 'Jan: 3' }, { x: 'Feb', y: 3.5,
text: 'Feb: 3.5' },
            { x: 'Mar', y: 7, text: 'Mar: 7' }, { x: 'Apr', y: 13.5,
text: 'Apr: 13.5' },
            { x: 'May', y: 19, text: 'May: 19' }, { x: 'Jun', y: 23.5,
text: 'Jun: 23.5' },
            { x: 'Jul', y: 26, text: 'Jul: 26' }, { x: 'Aug', y: 25,
text: 'Aug: 25' },
            { x: 'Sep', y: 21, text: 'Sep: 21' }, { x: 'Oct', y: 15,
text: 'Oct: 15' }],
        xName: 'x',
        yName: 'y'
    }
],
    legendSettings: {
        visible: true,
        reverse: true
    }
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

Legend Shape

To change the legend icon shape, use the `legendShape` property in the `series`. By default, legend icon shape is `seriesType`.

INDEX.JS

```

var piechart = new ej.charts.AccumulationChart({
  series: [
    {
      dataSource: [
        { x: 'Jan', y: 3, text: 'Jan: 3' }, { x: 'Feb', y: 3.5,
text: 'Feb: 3.5' },
        { x: 'Mar', y: 7, text: 'Mar: 7' }, { x: 'Apr', y: 13.5,
text: 'Apr: 13.5' },
        { x: 'May', y: 19, text: 'May: 19' }, { x: 'Jun', y: 23.5,
text: 'Jun: 23.5' },
        { x: 'Jul', y: 26, text: 'Jul: 26' }, { x: 'Aug', y: 25,
text: 'Aug: 25' },
        { x: 'Sep', y: 21, text: 'Sep: 21' }, { x: 'Oct', y: 15,
text: 'Oct: 15' }],
      xName: 'x',
      yName: 'y'
    }
  ],
  legendSettings: {
    position: 'Right',
    visible: true,
    height: '40',
    width: '160'
  }
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>

```

```
</body></html>
```

Legend Size

The legend size can be changed by using the `width` and `height` properties of the `legendSettings`.

INDEX.JS

```
var piechart = new ej.charts.AccumulationChart({
  series: [
    {
      dataSource: [
        { x: 'Jan', y: 3, text: 'Jan: 3' }, { x: 'Feb', y: 3.5,
text: 'Feb: 3.5' },
        { x: 'Mar', y: 7, text: 'Mar: 7' }, { x: 'Apr', y: 13.5,
text: 'Apr: 13.5' },
        { x: 'May', y: 19, text: 'May: 19' }, { x: 'Jun', y: 23.5,
text: 'Jun: 23.5' },
        { x: 'Jul', y: 26, text: 'Jul: 26' }, { x: 'Aug', y: 25,
text: 'Aug: 25' },
        { x: 'Sep', y: 21, text: 'Sep: 21' }, { x: 'Oct', y: 15,
text: 'Oct: 15' }],
      xName: 'x',
      yName: 'y'
    }
  ],
  legendSettings: {
    position: 'Right',
    visible: true,
    height: '40',
    width: '160'
  }
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
</body>
</html>
```

```

var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Legend Item Size

You can customize the size of the legend items by using the `shapeHeight` and `shapeWidth` properties.

INDEX.JS

```

var piechart = new ej.charts.AccumulationChart({
    series: [
        {
            dataSource: [
                { x: 'Jan', y: 3, text: 'Jan: 3' }, { x: 'Feb', y: 3.5,
text: 'Feb: 3.5' },
                { x: 'Mar', y: 7, text: 'Mar: 7' }, { x: 'Apr', y: 13.5,
text: 'Apr: 13.5' },
                { x: 'May', y: 19, text: 'May: 19' }, { x: 'Jun', y: 23.5,
text: 'Jun: 23.5' },
                { x: 'Jul', y: 26, text: 'Jul: 26' }, { x: 'Aug', y: 25,
text: 'Aug: 25' },
                { x: 'Sep', y: 21, text: 'Sep: 21' }, { x: 'Oct', y: 15,
text: 'Oct: 15' }],
            xName: 'x',
            yName: 'y'
        }
    ],
    legendSettings: {
        position: 'Right',
        visible: true,
        height: '40',
        width: '160'
    }
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>

```

```
<body>

    <div id="container">
        <div id="element"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Paging for Legend

Paging will be enabled by default, when the legend items exceeds the legend bounds. You can view the each legend item by navigating between the pages using the navigation buttons.

INDEX.JS

```
var piechart = new ej.charts.AccumulationChart({
    series: [
        {
            dataSource: [
                { x: 'Jan', y: 3, text: 'Jan: 3' }, { x: 'Feb', y: 3.5,
text: 'Feb: 3.5' },
                { x: 'Mar', y: 7, text: 'Mar: 7' }, { x: 'Apr', y: 13.5,
text: 'Apr: 13.5' },
                { x: 'May', y: 19, text: 'May: 19' }, { x: 'Jun', y: 23.5,
text: 'Jun: 23.5' },
                { x: 'Jul', y: 26, text: 'Jul: 26' }, { x: 'Aug', y: 25,
text: 'Aug: 25' },
                { x: 'Sep', y: 21, text: 'Sep: 21' }, { x: 'Oct', y: 15,
text: 'Oct: 15' }],
            xName: 'x',
            yName: 'y'
        }
    ],
    legendSettings: {
        position: 'Right',
        visible: true,
        height: '40',
        width: '160'
    }
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
```



```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Legend Text Wrap

When the legend text exceeds the container, the text can be wrapped by using `textWrap` Property. End user can also wrap the legend text based on the `maximumLabelWidth` property.

INDEX.JS

```

var piechart = new ej.charts.AccumulationChart({
    series: [
        {
            dataSource: [
                { 'x': 'Net-tution', y: 21, text: '21%' },
                { 'x': 'Private Gifts', y: 8, text: '8%' },
                { 'x': 'All Other', y: 9, text: '9%' },
                { 'x': 'Local Revenue', y: 4, text: '4%' },
                { 'x': 'State Revenue', y: 21, text: '21%' },
                { 'x': 'Federal Revenue', y: 16, text: '16%' },
                { 'x': 'Self-supporting Operations', y: 21, text: '21%' },
            ],
            xName: 'x',
            yName: 'y',
            startAngle: 0,
            endAngle: 360,
            innerRadius: '40%',
        }
    ],
    //Initializing Legend
    legendSettings: {
        visible: true,
        position: 'Right',
        height: '44%',
        width: '64%',
        textWrap: 'Wrap',
        maximumLabelWidth: 60,
    },
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Legend Title

You can set title for legend using `title` property in `legendSettings`. You can also customize the `fontStyle`, `size`, `fontWeight`,

`color`, `textAlignment`, `fontFamily`, `opacity` and `textOverflow` of legend title. `titlePosition` is used to set the legend position in `Top`, `Left` and `Right` position. `maximumTitleWidth` is used to set the width of the legend title. By default, it will be `100px`.

INDEX.JS

```

var piechart = new ej.charts.AccumulationChart({
  series: [
    {
      dataSource: [{ x: 'Jan', y: 13, text: 'Jan: 13' }, { x: 'Feb',
y: 13, text: 'Feb: 13' },
      { x: 'Mar', y: 17, text: 'Mar: 17' }, { x: 'Apr', y: 13.5, text:
'Apr: 13.5' }],
      xName: 'x',
      yName: 'y',
      type: 'Pie'
    }
  ],
  legendSettings: { visible: true, title: 'Months', position: 'Bottom' }
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Arrow Page Navigation

By default, the page number will be enabled while legend paging. Now, you can disable that page number and also you can get left and right arrows for page navigation. You have to set `false` value to `enablePages` to get this support.

INDEX.JS

```

var piechart = new ej.charts.AccumulationChart({
  series: [
    {
      dataSource: [{ x: 'Jan', y: 3 }, { x: 'Feb', y: 3.5 }, { x:
'Mar', y: 7 }, { x: 'Apr', y: 13.5 },
      { x: 'May', y: 19 }, { x: 'Jun', y: 23.5 }, { x: 'Jul', y: 26 },
{ x: 'Aug', y: 25 },
      { x: 'Sep', y: 21 }, { x: 'Oct', y: 15 }, { x: 'Nov', y: 15 },
{ x: 'Dec', y: 15 }],
      xName: 'x',
      yName: 'y',
    }
  ],
  legendSettings:{ width: '260px', height: '50px', enablePages: false,
position: 'Bottom' }
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Legend Item Padding

The [itemPadding](#) property can be used to adjust the space between the legend items.

INDEX.JS

```

var piechart = new ej.charts.AccumulationChart({
  series: [
    {
      dataSource: [{ x: 'Jan', y: 3 }, { x: 'Feb', y: 3.5 }, { x:
'Mar', y: 7 }, { x: 'Apr', y: 13.5 },
      { x: 'May', y: 19 }, { x: 'Jun', y: 23.5 }, { x: 'Jul', y: 26 },
{ x: 'Aug', y: 25 },
      { x: 'Sep', y: 21 }, { x: 'Oct', y: 15 }, { x: 'Nov', y: 15 },
{ x: 'Dec', y: 15 }],
      xName: 'x',
      yName: 'y',
    }
  ],
  legendSettings:{ width: '260px', height: '50px', position: 'Bottom',
legendPadding:30 }
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>

```

```

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Center label in ##Platform_Name## Accumulation chart control

Using [centerLabel](#) it is now possible to place a label at the center of a pie or doughnut chart. To configure the default text rendered on the center label for the pie and doughnut charts, use the [text](#) property in the [centerLabel](#).

INDEX.JS

```

var piechart = new ej.charts.AccumulationChart({
    series: [
        {
            dataSource: [
                { x: 'Chrome', y: 61.3, text: 'Chrome: 61.3%' },
                { x: 'Safari', y: 24.6, text: 'Safari: 24.6%' },
                { x: 'Edge', y: 5.0, text: 'Edge: 5.0%' },
                { x: 'Samsung Internet', y: 2.7, text: 'Samsung Internet: 2.7%' },
                { x: 'Firefox', y: 2.6, text: 'Firefox: 2.6%' },
                { x: 'Others', y: 3.6, text: 'Others: 3.6%' }],
            innerRadius: '65%',
            xName: 'x',
            yName: 'y'
        }
    ],
    centerLabel: {
        text : 'Mobile<br>Browsers<br>Statistics'
    }
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script id="Unemployment" type="text/x-template">
    <div id='templateWrap'>
      <table style="width:100%; border: 1px solid black;">
        <tr><th colspan="2" bgcolor="#00FFFF">Unemployment</th></tr>
        <tr><td bgcolor="#00FFFF">${x}:</td><td
bgcolor="#00FFFF">${y}</td></tr>
      </table>
    </div>
  </script>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Hover text

The default text in the center label can be changed when the mouse pointer hovers over the pie and doughnut charts slice using the [hoverTextFormat](#) property.

INDEX.JS

```

var piechart = new ej.charts.AccumulationChart({
  series: [
    {
      dataSource: [
        { x: 'Chrome', y: 61.3, text: 'Chrome: 61.3%' },
        { x: 'Safari', y: 24.6, text: 'Safari: 24.6%' },
        { x: 'Edge', y: 5.0, text: 'Edge: 5.0%' },
        { x: 'Samsung Internet', y: 2.7, text: 'Samsung Internet: 2.7%' }
      ],
      { x: 'Firefox', y: 2.6, text: 'Firefox: 2.6%' },
      { x: 'Others', y: 3.6, text: 'Others: 3.6%' }
    ],
    innerRadius: '65%',
    xName: 'x',
    yName: 'y'
  ]
});

```

```

    }
    ],
    centerLabel:{
        text : 'Mobile<br>Browsers<br>Statistics',
        hoverTextFormat: '${point.x} <br> Browser Share <br> ${point.y}%'
    }
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
    <script id="Unemployment" type="text/x-template">
        <div id='templateWrap'>
            <table style="width:100%; border: 1px solid black;">
                <tr><th colspan="2" bgcolor="#00FFFF">Unemployment</th></tr>
                <tr><td bgcolor="#00FFFF">${x}</td><td
                bgcolor="#00FFFF">${y}</td></tr>
            </table>
        </div>
    </script>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

Customization

Customize the center label text using the [textStyle](#) property.

INDEX.JS

```

var piechart = new ej.charts.AccumulationChart({
    series: [
        {
            dataSource: [ { x: 'Chrome', y: 61.3, text: 'Chrome: 61.3%' },

```

```

        { x: 'Safari', y: 24.6, text: 'Safari: 24.6%' },
        { x: 'Edge', y: 5.0, text: 'Edge: 5.0%' },
        { x: 'Samsung Internet', y: 2.7, text: 'Samsung Internet: 2.7%'
    },
    [
        { x: 'Firefox', y: 2.6, text: 'Firefox: 2.6%' },
        { x: 'Others', y: 3.6, text: 'Others: 3.6%' }],
    innerRadius: '65%',
    xName: 'x',
    yName: 'y'
    }
],
centerLabel:{
    text : 'Mobile<br>Browsers<br>Statistics',
    textStyle: {
        fontWeight: 600
    }
}
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
    <script id="Unemployment" type="text/x-template">
        <div id='templateWrap'>
            <table style="width:100%; border: 1px solid black;">
                <tr><th colspan="2" bgcolor="#00FFFF">Unemployment</th></tr>
                <tr><td bgcolor="#00FFFF">${x}</td><td
            bgcolor="#00FFFF">${y}</td></tr>
            </table>
        </div>
    </script>
    <script>
    var ele = document.getElementById('container');
    if(ele) {
        ele.style.visibility = "visible";
    }
    </script>
    <script src="index.js" type="text/javascript"></script>

```



```
</body></html>
```

Title and sub title in ##Platform_Name## Accumulation chart control

Accumulation Chart can be given a title using [title](#) property, to show the information about the data plotted.

INDEX.JS

```
var piechart = new ej.charts.AccumulationChart({
  series: [
    {
      dataSource: [
        { x: 'Saudi Arabia', y: 58, text: '58%' },
        { x: 'Persian Gulf', y: 15, text: '15%' },
        { x: 'Canada', y: 13, text: '13%' },
        { x: 'Venezuela', y: 8, text: '8%' },
        { x: 'Mexico', y: 3, text: '3%' },
        { x: 'Russia', y: 2, text: '2%' },
        { x: 'Miscellaneous', y: 1, text: '1%' }
      ],
      xName: 'x', yName: 'y',
      dataLabel: {
        visible: true,
        name: 'text',
        font: { color: 'white' }
      },
    },
  ],
  legendSettings: {
    visible: false,
  },
  title: 'Oil and other liquid imports in USA',
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
```

```
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Title Customization

Accumulation Chart can be customizing a title using [titleStyle](#) property.

INDEX.JS

```
var piechart = new ej.charts.AccumulationChart({
    series: [
        {
            dataSource: [
                { x: 'Saudi Arabia', y: 58, text: '58%' },
                { x: 'Persian Gulf', y: 15, text: '15%' },
                { x: 'Canada', y: 13, text: '13%' },
                { x: 'Venezuela', y: 8, text: '8%' },
                { x: 'Mexico', y: 3, text: '3%' },
                { x: 'Russia', y: 2, text: '2%' },
                { x: 'Miscellaneous', y: 1, text: '1%' }
            ],
            xName: 'x', yName: 'y',
            dataLabel: {
                visible: true,
                name: 'text',
                font: { color: 'white' }
            },
        },
    ],
    legendSettings: {
        visible: false,
    },
    title: 'Oil and other liquid imports in USA',
    titleStyle: {
        fontFamily: 'Arial',
        fontStyle: 'italic',
        fontWeight: 'regular',
        color: '#E27F2D',
        size: '23px'
    }
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
```

```

<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

SubTitle

Accumulation Chart can be given a subtitle using [subTitle](#) property, to show the information about the data plotted.

INDEX.JS

```

var piechart = new ej.charts.AccumulationChart({
    series: [
        {
            dataSource: [
                { x: 'Saudi Arabia', y: 58, text: '58%' },
                { x: 'Persian Gulf', y: 15, text: '15%' },
                { x: 'Canada', y: 13, text: '13%' },
                { x: 'Venezuela', y: 8, text: '8%' },
                { x: 'Mexico', y: 3, text: '3%' },
                { x: 'Russia', y: 2, text: '2%' },
                { x: 'Miscellaneous', y: 1, text: '1%' }
            ],
            xName: 'x', yName: 'y',
            dataLabel: {
                visible: true,
                name: 'text',
                font: { color: 'white' }
            },
        },
    ],
    legendSettings: {
        visible: false,
    },
    title: 'Oil and other liquid imports in USA',
    subTitle : 'In the year 2014 - 2015'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

SubTitle Customization

Accumulation Chart can be customizing a subtitle using [subTitleStyle](#) property, to show the information about the data plotted.

INDEX.JS

```

var piechart = new ej.charts.AccumulationChart({
  series: [
    {
      dataSource: [
        { x: 'Saudi Arabia', y: 58, text: '58%' },
        { x: 'Persian Gulf', y: 15, text: '15%' },
        { x: 'Canada', y: 13, text: '13%' },
        { x: 'Venezuela', y: 8, text: '8%' },
        { x: 'Mexico', y: 3, text: '3%' },
        { x: 'Russia', y: 2, text: '2%' },
        { x: 'Miscellaneous', y: 1, text: '1%' }
      ],
      xName: 'x', yName: 'y',
      dataLabel: {
        visible: true,
        name: 'text',
        font: { color: 'white' }
      },
    },
  ],
});

```

```

legendSettings: {
    visible: false,
},
title: 'Oil and other liquid imports in USA',
subTitle : 'In the year 2014 - 2015'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

Chart print in ##Platform_Name## Accumulation chart control

Print

The rendered chart can be printed directly from the browser by calling the public method print.

INDEX.JS

```

var piechart = new ej.charts.AccumulationChart({
    series: [
        {
            dataSource: [{ x: 'Jan', y: 3 }, { x: 'Feb', y: 3.5 }, { x:
'Mar', y: 7 }, { x: 'Apr', y: 13.5 },
            { x: 'May', y: 19 }, { x: 'Jun', y: 23.5 }, { x: 'Jul', y: 26 },
{ x: 'Aug', y: 25 },
            { x: 'Sep', y: 21 }, { x: 'Oct', y: 15 }],
            radius: '100%',
            xName: 'x',
            yName: 'y'
        }
    ]
});

```

```

    ]
  }, '#element');
  document.getElementById('print').onclick = () => {
    chart.print();
  };

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element" style="float: left "></div>
    <button id="print" type="button" width="15%" style="float:
right">Print</button>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Export

The rendered chart can be exported to **Image**(jpeg or png) or **SVG** or **PDF** format by using the export method.

Input parameters for this method are **Export Type** for **format** and **fileName** of result.

INDEX.JS

```

var piechart = new ej.charts.AccumulationChart({
  series: [
    {
      dataSource: [{ x: 'Jan', y: 3 }, { x: 'Feb', y: 3.5 }, { x:
'Mar', y: 7 }, { x: 'Apr', y: 13.5 },
      { x: 'May', y: 19 }, { x: 'Jun', y: 23.5 }, { x: 'Jul', y: 26 },
{ x: 'Aug', y: 25 },
      { x: 'Sep', y: 21 }, { x: 'Oct', y: 15 }],
      radius: '100%',

```

```

        xName: 'x',
        yName: 'y'
    }
]
}, '#element');
document.getElementById('print').onclick = () => {
    chart.export('PNG', 'result');
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element" style="float: left "></div>
    <div id="element1" style="float: left "></div>
    <button id="print" type="button" width="15%" style="float:
right">Export</button>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Accessibility in ##Platform_Name## Accumulation chart control

The Accumulation chart control followed the accessibility guidelines and standards, including [ADA](#), [Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Accumulation chart control is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support | |

```

| Section 508 Support |  |

| Screen Reader Support |  |

| Right-To-Left Support |  |

| Color Contrast |  |

| Mobile Device Support |  |

| Keyboard Navigation Support |  |

| Accessibility Checker Validation |  |

| Axe-core Accessibility Validation |  |

<style>

.post .post-content img {
display: inline-block;
margin: 0.5em 0;
}

</style>

<div> - All
features of the control meet the requirement.</div>

<div> - Some features of the control do not meet the requirement.</div>

<div> - The control does not meet the requirement.</div>

```

WAI-ARIA attributes

The Accumulation chart control followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Accumulation chart control:

- `img` (role)
- `button` (role)
- `region` (role)
- `aria-label` (attribute)
- `aria-hidden` (attribute)
- `aria-pressed` (attribute)

Keyboard interaction

The Accumulation chart control followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Accumulation chart control.

- | Press | To do this |
- | --- | --- |
- | Alt + J | Moves the focus to the Accumulation chart element. |
- | Tab | Moves the focus to the next element in the Accumulation chart. |
- | Shift + Tab | Moves the focus to the previous element in the Accumulation chart. |
- | Down Arrow | Moves the focus to the data point left side from the selected point. |
- | Up Arrow | Moves the focus to the data point right side from the selected point. |
- | Down/Left Arrow | Moves the focus to the legend left side from the selected legend. |
- | Up/Right Arrow | Moves the focus to the legend right side from the selected legend. |
- | Enter/Space | Toggles the visibility of the corresponding series. |
- | Ctrl + P | Prints the Accumulation chart. |

Ensuring accessibility

The Accumulation chart control's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Accumulation chart control is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Accumulation chart control with accessibility tools.

See also

- [Accessibility in Syncfusion ##Platform_Name## controls](#)

Ej1 api migration in ##Platform_Name## Accumulation chart control

This article describes the API migration process of Chart component from Essential JS 1 to Essential JS 2.

Accumulation Chart

<!-- markdownlint-disable MD033 -->

Behaviour	API in Essential JS 1	API in Essential JS 2
annotations	Property:annotations\$("#chart").ejChart({ annotations: [{}]});	Property:annotationslet pie: AccumulationChart = new AccumulationChart({annotations: [{}]});pie.appendTo('#chart');
background	Property:background\$("#chart").ejChart({ background: '#DDCCEE'});	Property:backgroundlet pie: AccumulationChart = new AccumulationChart({

		<code>background: 'DDCCEE');pie.appendTo('#chart');</code>
border	<code>Property:border\$("#chart").ejChart({ border: { color: 'red' , width: 2, opacity: 0.5}});</code>	<code>Property:borderlet pie: AccumulationChart = new AccumulationChart({ border: { color: 'red', width: 2}});pie.appendTo('#chart');</code>
dataSource	Not applicable	<code>Property:borderlet pie: AccumulationChart = new AccumulationChart({ dataSource: []);});pie.appendTo('#chart');</code>
Animation after legend click	Not applicable	<code>Property:enableAnimationlet pie: AccumulationChart = new AccumulationChart({ enableAnimation: true});pie.appendTo('#chart');</code>
Persisting component's state between page reloads.	Not applicable	<code>Property:enablePersistancellet pie: AccumulationChart = new AccumulationChart({ enablePersistancel: true});pie.appendTo('#chart');</code>
Enabling smart labels	<code>Property:series.enableSmartLabels\$("#chart").ejChart({ series: [{ enableSmartLabels: true}]});</code>	<code>Property:enableSmartLabelslet pie: AccumulationChart = new AccumulationChart({ enableSmartLabels: true});pie.appendTo('#chart');</code>
Height of Chart	<code>Property:size.height\$("#chart").ejChart({ size: { height: '400' }});</code>	<code>Property:heightlet pie: AccumulationChart = new AccumulationChart({ height: '400'});pie.appendTo('#chart');</code>
Multi selection	<code>Property:selectionSettings.type\$("#chart").ejChart({ selectionSettings: { type: 'multiple' }});</code>	<code>Property:isMultiSelectlet pie: AccumulationChart = new AccumulationChart({ isMultiSelect: true});pie.appendTo('#chart');</code>
legend Settings	<code>Property:legend\$("#chart").ejChart({ legend: { }});</code>	<code>Property:legendSettingslet pie: AccumulationChart = new AccumulationChart({ legendSettings: { }});pie.appendTo('#chart');</code>

Margin for the chart	Property:margin\$("#chart").ejChart({ margin: { top: 20, bottom: 23, right: 15, left: 10 }});	Property:marginlet pie: AccumulationChart = new AccumulationChart({ margin: { top: 20, bottom: 23, right: 15, left: 10 }});pie.appendTo('#chart');
SelectedDataIndexes	Property:selectedDataPointIndexes \$("#chart").ejChart({ selectedDataPointIndexes : [{}]});	Property:selectedDataIndexes let pie: AccumulationChart = new AccumulationChart({ selectedDataIndexes : [{ series: 0, point: 1}]});pie.appendTo('#chart') ;
Selection Mode	Property:selectionSettings.mode\$("#chart"). ejChart({ selectionSettings: { mode: 'Point'}});	Property:selectionMode let pie: AccumulationChart = new AccumulationChart({ selectionMode : 'Point'});pie.appendTo('#chart');
Series	Property:series\$("#chart").ejChart({ series: []});	Property:serieslet pie: AccumulationChart = new AccumulationChart({ series: []});pie.appendTo('#chart');
Title text	Property:title.text\$("#chart").ejChart({ title: { text: 'Pie Chart' }});	Property:titlelet pie: AccumulationChart = new AccumulationChart({ title: 'Pie Chart'});pie.appendTo('#chart');
Title Styles	Property:title\$("#chart").ejChart({ title: { text: 'Pie Chart' }});	Property:titleStylelet pie: AccumulationChart = new AccumulationChart({ titleStyle: { fontFamily: 'SegoeUI'}});pie.appendTo('#chart');
Sub Title text	Property:subTitle.text\$("#chart").ejChart({ subTitle: { text: 'Pie Chart' }});	Property:subTitlelet pie: AccumulationChart = new AccumulationChart({ subTitle: 'Pie Chart'});pie.appendTo('#chart');
Sub title Styles	Property:title\$("#chart").ejChart({ title: { text: 'Pie Chart' }});	Property:subTitleStylelet pie: AccumulationChart = new AccumulationChart({ subTitleStyle: { fontFamily: 'SegoeUI'}});pie.appendTo('#chart');

tooltip	Property:series.toolTip\$("#chart").ejChart({ series: [{ tooltip: { } }] });	Property:tooltiplet pie: AccumulationChart = new AccumulationChart({ tooltip: { } });pie.appendTo('#chart');
Width of Chart	Property:size.width\$("#chart").ejChart({ size: { width: '400' } });	Property:widthlet pie: AccumulationChart = new AccumulationChart({ width: '400' });pie.appendTo('#chart');

Annotation

<!-- markdownlint-disable MD033 -->

Behaviour	API in Essential JS 1	API in Essential JS 2
annotations	Property:annotations\$("#chart").ejChart({ annotations: [{}];});	Property:annotationslet pie: AccumulationChart = new AccumulationChart({ annotations: [{}];});pie.appendTo('#chart');
content	Property:annotations\$("#chart").ejChart({ annotations: [{content: 'Pie Chart'}];});	Property:annotationslet pie: AccumulationChart = new AccumulationChart({ annotations: [{content: 'Pie Chart'}];});pie.appendTo('#chart');
coordinateUnits	Property:annotations\$("#chart").ejChart({ annotations: [{coordinateUnit: 'Pixel'}];});	Property:annotationslet pie: AccumulationChart = new AccumulationChart({ annotations: [{coordinateUnit: 'Pixel'}];});pie.appendTo('#chart');
description	Not Applicable	Property:descriptionlet pie: AccumulationChart = new AccumulationChart({ annotations: [{description: 'Pixel'}];});pie.appendTo('#chart');
horizontalAlignment for annotation	Property:annotations.horizontalAlignment\$("#container").ejChart({ annotations :[{ horizontalAlignment : "middle" }] });	Property:annotations.horizontalAlignmentlet chart: AccumulationChart = new AccumulationChart({ annotations: [{ horizontalAlignment: 'Center' }] });chart.appendTo('#chart');

margin for annotation	Property:annotations.margin \$("#container").ejChart({ annotations :[{ margin { right: 5, left: 5, top: 5, bottom: 5}}]});	Not applicable
Opacity for annotation	Property:annotations.opacity \$("#container").ejChart({ annotations :[{ opacity: 0.4 }]});	Not applicable
Region for annotation with respect to chart or series	Property:annotations.region \$("#container").ejChart({ annotations :[{ region : "chart"}]});	Property:annotations.region let chart: AccumulationChart = new AccumulationChart({ annotations : [{ region: 'Chart'}]}); chart.appendTo('#chart');
verticalAlignment for annotation	Property:annotations.verticalAlignment \$("#container").ejChart({ annotations :[{ verticalAlignment : "middle"}]});	Property:annotations.verticalAlignment let chart: AccumulationChart = new AccumulationChart({ annotations : [{ verticalAlignment: 'Center'}]}); chart.appendTo('#chart');
Visibility of annotations	Property:annotations.visible \$("#container").ejChart({ annotations :[{ visible: true }]});	Not applicable
X offset for annotation	Property:annotations.x \$("#container").ejChart({ annotations :[{ x : "100"}]});	Property:annotations.x let chart: AccumulationChart = new AccumulationChart({ annotations : [{ x: '100'}]}); chart.appendTo('#chart');
Y offset for annotation	Property:annotations.y \$("#container").ejChart({ annotations :[{ y : "100"}]});	Property:annotations.y let chart: AccumulationChart = new AccumulationChart({ annotations : [{ y: '100'}]}); chart.appendTo('#chart');

Series

<!-- markdownlint-disable MD033 -->

Behaviour	API in Essential JS 1	API in Essential JS 2
series	Property:series \$("#chart").ejChart({ series: [{ }]});	Property:series let pie: AccumulationChart = new AccumulationChart({ series: [{ }]}); pie.appendTo('#chart');

animation for series	Property:enableAnimation\$("#chart").ejChart({ series: [{ enableAnimation: true}]});	Property:animation.enablelet pie: AccumulationChart = new AccumulationChart({ series: [{ animation: { enable: true } }]});pie.appendTo('#chart');
animation duration for series	Property:animationDuration\$("#chart").ejChart({ series: [{ animationDuration: 1000}]});	Property:animation.durationlet pie: AccumulationChart = new AccumulationChart({ series: [{ animation: { duration: 1000 } }]});pie.appendTo('#chart');
animation delay for series	Not Applicable	Property:animation.durationlet pie: AccumulationChart = new AccumulationChart({ series: [{ animation: { delay: 100 } }]});pie.appendTo('#chart');
Border for series	Property:border\$("#chart").ejChart({ series: [{ border: { color: 'pink', width: 2, dashArray: '10,4' } }]});	Property:borderlet pie: AccumulationChart = new AccumulationChart({ series: [{ border: { color: 'red', width: 2 } }]});pie.appendTo('#chart');
DataLabel for series	Property:dataLabel\$("#chart").ejChart({ series: [{ dataLabel: {} }]});	Property:borderlet pie: AccumulationChart = new AccumulationChart({ series: [{ dataLabel: { } }]});pie.appendTo('#chart');
DataSource for series	Property:dataSource\$("#chart").ejChart({ series: [{ dataSource: [{ }] }]});	Property:dataSourcelet pie: AccumulationChart = new AccumulationChart({ series: [{ dataSource: [{ }] }]});pie.appendTo('#chart');
enableTooltip for series	Property:tooltip.visible\$("#chart").ejChart({ series: [{ tooltip: { visible: true } }]});	Property:enableTooltiplet pie: AccumulationChart = new AccumulationChart({ series: [{ enableTooltip: true }] }]);pie.appendTo('#chart');
start angle	Property:startAngle\$("#chart").ejChart({ series: [{ startAngle: 80 }]});	Property:startAnglelet pie: AccumulationChart = new AccumulationChart({ series: [{ startAngle: 90

		<code>}};});pie.appendTo('#chart');</code>
end angle	<code>Property:endAngle\$("#chart").ejChart({ series: [{ endAngle: 80 }]});</code>	<code>Property:endAnglelet pie: AccumulationChart = new AccumulationChart({ series: [{ endAngle: 90 }]});pie.appendTo('#chart');</code>
explode	<code>Property:explode\$("#chart").ejChart({ series: [{ explode: true }]});</code>	<code>Property:explodelet pie: AccumulationChart = new AccumulationChart({ series: [{ explode: true }]});pie.appendTo('#chart');</code>
explodeAll	<code>Property:explodeAll\$("#chart").ejChart({ series: [{ explodeAll: true }]});</code>	<code>Property:explodeAlllet pie: AccumulationChart = new AccumulationChart({ series: [{ explodeAll: true }]});pie.appendTo('#chart');</code>
explodeIndex	<code>Property:explode\$("#chart").ejChart({ series: [{ explodeIndex: 0 }]});</code>	<code>Property:explodeIndexlet pie: AccumulationChart = new AccumulationChart({ series: [{ explodeIndex: 1 }]});pie.appendTo('#chart');</code>
explodeOffset	<code>Property:explodeOffset\$("#chart").ejChart({ series: [{ explodeOffset: '30%' }]});</code>	<code>Property:explodeOffsetlet pie: AccumulationChart = new AccumulationChart({ series: [{ explodeOffset: '30%' }]});pie.appendTo('#chart');</code>
gapRatio	<code>Property:gapRatio\$("#chart").ejChart({ series: [{ gapRatio: 0.6 }]});</code>	<code>Property:gapRatiolet pie: AccumulationChart = new AccumulationChart({ series: [{ gapRatio: 0.6 }]});pie.appendTo('#chart');</code>
gapWidth	<code>Property:gapWidth\$("#chart").ejChart({ series: [{ gapWidth: 0.6 }]});</code>	Not Applicable
inner radius of the accumulation chart	<code>Property:innerRadius\$("#chart").ejChart({ series: [{ innerRadius : '30%' }]});</code>	<code>Property:innerRadiuslet pie: AccumulationChart = new AccumulationChart({ series: [{ innerRadius : '30%' }]});pie.appendTo('#chart');</code>

Legend shape of the series	Not applicable	Property:legendShape let pie: AccumulationChart = new AccumulationChart({ series: [{ legendShape : 'Rectangle' }]});pie.appendTo('#chart');
name of the series	Property:name \$("#chart").ejChart({ series: [{ name : '30%' }]});	Property:name let pie: AccumulationChart = new AccumulationChart({ series: [{ name : '30%' }]});pie.appendTo('#chart');
neck height for funnel series	Property:neckHeight \$("#chart").ejChart({ series: [{ neckHeight : '30%' }]});	Property:neckHeight let pie: AccumulationChart = new AccumulationChart({ series: [{ neckHeight : '30%' }]});pie.appendTo('#chart');
neck width for funnel series	Property:neckWidth \$("#chart").ejChart({ series: [{ neckWidth : '30%' }]});	Property:neckWidth let pie: AccumulationChart = new AccumulationChart({ series: [{ neckWidth : '30%' }]});pie.appendTo('#chart');
opacity for series	Property:opacity \$("#chart").ejChart({ series: [{ opacity : 0.4 }]});	Property:opacity let pie: AccumulationChart = new AccumulationChart({ series: [{ opacity : 0.5 }]});pie.appendTo('#chart');
palettes for series	Property:palette \$("#chart").ejChart({ series: [{ palette : [] }]});	Property:palettes let pie: AccumulationChart = new AccumulationChart({ series: [{ palettes : [] }]});pie.appendTo('#chart');
point color mapping name for series	Property:pointColorMappingName \$("#chart").ejChart({ series: [{ pointColorMappingName : 'color' }]});	Property:pointColorMapping let pie: AccumulationChart = new AccumulationChart({ series: [{ pointColorMappingName : 'color' }]});pie.appendTo('#chart');
Mode of pyramid series	Property:pyramidMode \$("#chart").ejChart({ series: [{ pyramidMode : 'Surface' }]});	Property:pyramidMode let pie: AccumulationChart = new AccumulationChart({ series: [{ pyramidMode : 'Linear' }]});

		<code>});});pie.appendTo('#chart');</code>
query for datasource for series	<code>Property:pyramidMode\$("#chart").ejChart({ series: [{ query : ' ' }]});</code>	<code>Property:querylet pie: AccumulationChart = new AccumulationChart({ series: [{ query : ' ' }]});pie.appendTo('#chart');</code>
Radius of Pie series	<code>Property:pieCoefficient\$("#chart").ejChart({ series: [{ pieCoefficient : 0.5 }]});</code>	<code>Property:radiuslet pie: AccumulationChart = new AccumulationChart({ series: [{ radius: '50%' }]});pie.appendTo('#chart');</code>
Selection Style of Accumulation chart	Not applicable	<code>Property:selectionStylelet pie: AccumulationChart = new AccumulationChart({ series: [{ selectionStyle: ' ' }]});pie.appendTo('#chart');</code>
tooltip Mapping name	Not applicable	<code>Property:tooltipMappingNamelet pie: AccumulationChart = new AccumulationChart({ series: [{ tooltipMappingName: ' ' }]});pie.appendTo('#chart');</code>
Type of series	<code>Property:type\$("#chart").ejChart({ series: [{ type : 'Pie' }]});</code>	<code>Property:typetlet pie: AccumulationChart = new AccumulationChart({ series: [{ type: 'Pie' }]});pie.appendTo('#chart');</code>
Name of the property in the datasource that contains x value for the series.	<code>Property:xName\$("#chart").ejChart({ series: [{ xName : 'x' }]});</code>	<code>Property:xNamelet chart: AccumulationChart = new AccumulationChart({ series: [{ xName: 'x' }]});chart.appendTo('#chart');</code>
Name of the property in the datasource that	<code>Property:yName\$("#chart").ejChart({ series: [{ yName : 'x' }]});</code>	<code>Property:yNamelet chart: AccumulationChart = new AccumulationChart({ series: [{ yName: 'x' }]});chart.appendTo('#chart');</code>

contains x value for the series.		
Name of the property in the datasource that contains x value for the series.	Property:yName\$("#chart").ejChart({ series: [{ yName : 'x' }]});	Property:yNamelet chart: AccumulationChart = new AccumulationChart({ series: [{ yName: 'x' }]});chart.appendTo('#chart');
Width of funnel series	Property:funnelWidth\$("#chart").ejChart({ series: [{ funnelWidth : '100' }]});	Property:widthlet chart: AccumulationChart = new AccumulationChart({ series: [{ width: '100' }]});chart.appendTo('#chart');
Grouping	Not Applicable	Property:groupTolet chart: AccumulationChart = new AccumulationChart({ series: [{ groupTo: '100' }]});chart.appendTo('#chart');
GroupMode	Not Applicable	Property:groupModelet chart: AccumulationChart = new AccumulationChart({ series: [{ groupMode: 'Point' }]});chart.appendTo('#chart');

DataLabel

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
dataLabel	Property:series.marker.dataLabel\$("#chart").ejChart({ series: [{ marker: { dataLabel : { visible: true }}}]});	Property:series.dataLabellet pie: AccumulationChart = new AccumulationChart({ series: { dataLabel : { visible: true }}});pie.appendTo('#chart');
border of dataLabel	Property:series.marker.dataLabel.border\$("#chart").ejChart({ series: [{ marker: { dataLabel : { border: { color: 'red', width: 2 }}} }]});	Property:borderlet pie: AccumulationChart = new AccumulationChart({ series: { dataLabel : { border: { color: 'red', width:

		2}}}});pie.appendTo('#chart');
connect or style for dataLabel connector or line	Property:connectorLine\$("#chart").ejChart({ series: [{ marker: { dataLabel :{ connectorLine: { type: 'Curve', width: 2 }}}}]});	Property:connectorStylelet pie: AccumulationChart = new AccumulationChart({ series: { dataLabel : { connectorStyle: { type: 'Curve', width: 2 }}}});pie.appendTo('#chart') ;
Fill for dataLabel	Property:fill\$("#chart").ejChart({ series: [{ marker: { dataLabel :{ fill: 'red' }}}]});	Property:filllet pie: AccumulationChart = new AccumulationChart({ series: { dataLabel : { fill: 'pink' }}});pie.appendTo('#chart');
font for dataLabel	Property:font\$("#chart").ejChart({ series: [{ marker: { dataLabel :{ font: { } }}}]});	Property:fontlet pie: AccumulationChart = new AccumulationChart({ series: { dataLabel : { font: { }}}});pie.appendTo('#chart') ;
font for dataLabel	Property:font\$("#chart").ejChart({ series: [{ marker: { dataLabel :{ font: { } }}}]});	Property:fontlet pie: AccumulationChart = new AccumulationChart({ series: { dataLabel : { font: { }}}});pie.appendTo('#chart') ;
position	Property:position\$("#chart").ejChart({ series: [{ marker: { dataLabel :{ position: 'Inside' }}}]});	Property:fontlet pie: AccumulationChart = new AccumulationChart({ series: { dataLabel : { position: 'Outside' }}}});pie.appendTo('#chart');
Rounde d corner radius X	Not Applicable	Property:dataLabel.rxlet chart: AccumulationChart = new AccumulationChart({ series: [{dataLabel: { rx: 10 } }}]);chart.appendTo('#chart') ;
Rounde d corner radius Y	Not Applicable	Property:dataLabel.rylet chart: AccumulationChart = new AccumulationChart({ series: [{dataLabel: { ry: 10 } }}]);chart.appendTo('#chart') ;

HTML template in dataLabel	Property:dataLabel.template <pre>\$("#chart").ejChart({ series: [{ marker: { dataLabel: { template: 'Chart' } } }]});</pre>	Property:dataLabel.template <pre>let chart: AccumulationChart = new AccumulationChart({ series: [{dataLabel: { template: 'Chart' } }]}); chart.appendTo('#chart');</pre>
----------------------------	--	--

Legend

<!-- markdownlint-disable MD033 -->

Behaviour	API in Essential JS 1	API in Essential JS 2
Default legend	Property:visible <pre>\$("#container").ejChart({ legend: { visible: true } });</pre>	Property:visible <pre>let chart: AccumulationChart = new AccumulationChart({ legendSettings: { visible: true } });</pre>
Legend height	Property:size.height <pre>\$("#container").ejChart({ legend: { size : { height: 50 } } });</pre>	Property:height <pre>let chart: AccumulationChart = new AccumulationChart({ legendSettings: { height: '30' } });</pre>
Legend width	Property:size.width <pre>\$("#container").ejChart({ legend: { size: { width: 20 } } });</pre>	Property:width <pre>let chart: AccumulationChart = new AccumulationChart({ legendSettings: { width: '30' } });</pre>
Legend location in chart	Property:location <pre>\$("#container").ejChart({ legend: { location: { x: 3, y: 45 } } });</pre>	Property:height <pre>let chart: AccumulationChart = new AccumulationChart({ legendSettings: { location: { x: 3, y: 45 } } });</pre>
Legend position in chart	Property:position <pre>\$("#container").ejChart({ legend: { position: 'top' } });</pre>	Property:position <pre>let chart: AccumulationChart = new AccumulationChart({ legendSettings: { position: 'Top' } });</pre>
Legend padding	Not applicable	Property:padding <pre>let chart: AccumulationChart = new AccumulationChart({ legendSettings: { padding: 8 } });</pre>

Legend alignment	Property:position <pre>\$("#container").ejChart({ legend: { alignment: 'center' } });</pre>	Property:position <pre>let chart: AccumulationChart = new AccumulationChart({ legendSettings: { alignment: 'Center' } });</pre>
text style for legend	Property:font <pre>\$("#container").ejChart({ legend: { font: { fontFamily: '', fontWeight: '400', fontStyle: 'italic', size: '12' } }});</pre>	Property:textStyle <pre>let chart: AccumulationChart = new AccumulationChart({ legendSettings: { textStyle: { size: '12' , color: 'red', fontFamily: 'Italic', fontWeight: '400', fontStyle: 'Normal', opacity: 1, textAlignment: 'Center', textOverflow: 'Trim' } }});</pre>
shape height of legend	Property:itemStyle.height <pre>\$("#container").ejChart({ legend: { itemStyle: { height: 20 } } });</pre>	Property:shapeHeight <pre>let chart: AccumulationChart = new AccumulationChart({ legendSettings: { shapeHeight: 20 } });</pre>
shape width of legend	Property:itemStyle.width <pre>\$("#container").ejChart({ legend: { itemStyle: { width: 20 } } });</pre>	Property:shapeWidth <pre>let chart: AccumulationChart = new AccumulationChart({ legendSettings: { shapeWidth: 20 } });</pre>
shape width of legend	Property:itemStyle.width <pre>\$("#container").ejChart({ legend: { itemStyle: { width: 20 } } });</pre>	Property:shapeWidth <pre>let chart: AccumulationChart = new AccumulationChart({ legendSettings: { shapeWidth: 20 } });</pre>
shape border of legend	Property:itemStyle.border <pre>\$("#container").ejChart({ legend: { itemStyle: { border: { width: 2, color: 'red' } } } });</pre>	Not Applicable
shape padding of legend	Property:itemPadding <pre>\$("#container").ejChart({ legend: { itemPadding: 10 } });</pre>	Property:shapePadding <pre>let chart: AccumulationChart = new AccumulationChart({ legendSettings: { shapePadding: 20 } });</pre>
Background of legend	Property:background <pre>\$("#container").ejChart({ legend: { background: 'transparent' } });</pre>	Property:backgorund <pre>let chart: AccumulationChart = new AccumulationChart({ legendSettings: { background: 'transparent' }});</pre>
Opacity of legend	Property:opacity <pre>\$("#container").ejChart({ legend: { opacity: 0.3 } });</pre>	Property:opacity <pre>let chart: AccumulationChart = new AccumulationChart({</pre>

		legendSettings: { opacity: 0.4 }});
Toggle visibility of series while legend click	Property:toggleSeriesVisibility\$("#container").ejChart({ legend: { toggleSeriesVisibility: true }});	Property:toggleVisibility let chart: AccumulationChart = new AccumulationChart({ legendSettings: { toggleVisibility: true }});
Title for legend	Property:title\$("#container").ejChart({ legend: { title: { text: 'LegendTitle', font: { }, textAlignment: 'middle' } }});	Not applicable
Text Overflow for legend	Property:title\$("#container").ejChart({ legend: { textOverflow: 'trim' }});	Property:textStyle.textOverflow let chart: new AccumulationChart({ legend: { text: { textOverflow: 'trim' } }});
Text width for legend while setting text overflow	Property:textWidth\$("#container").ejChart({ legend: { textWidth: 20 }});	Not applicable
Scroll bar for legend	Property:enableScrollBar\$("#container").ejChart({ legend: { enableScrollBar: true }});	Not applicable
Row count for legend	Property:rowCount\$("#container").ejChart({ legend: { rowCount: 2 }});	Not applicable
Column count for legend	Property:columnCount\$("#container").ejChart({ legend: { columnCount: 2 }});	Not applicable
Color for legend items	Property:fill\$("#container").ejChart({ legend: { fill: '#EEFFCC' }});	Not applicable

Methods

<!-- markdownlint-disable MD033 -->

Behaviour	API in Essential JS 1	API in Essential JS 2
animation for series	Property:chart.animate\$("#chart").ejChart({ animate: () { }});	Not applicable

Redraw for chart	Property:chart.redraw\$("#chart").ejChart({ redraw: () { } });	Property:chart.refresh()let chart: AccumulationChart = new AccumulationChart({});chart.appendTo('#chart');chart.width = '400';chart.refresh();
Export	Property:chart.export()\$("#chart").ejChart({ export: () { } });	Property:chart.export()let chart: AccumulationChart = new AccumulationChart({});chart.export('JPEG', 'chart');chart.appendTo('#chart');
Print	Property:chart.print()\$("#chart").ejChart({ print: () { } });	Property:chart.print()let chart: AccumulationChart = new AccumulationChart({});chart.print('chart');chart.appendTo('#chart');
AddSeries	Not Applicable	Property:chart.addSeries()let chart: AccumulationChart = new AccumulationChart({});chart.appendTo('#chart');chart.addSeries();
RemoveSeries	Not Applicable	Property:chart.removeSeries()let chart: AccumulationChart = new AccumulationChart({});chart.appendTo('#chart');chart.removeSeries();

Events

<!-- markdownlint-disable MD033 -->

Behaviour	API in Essential JS 1	API in Essential JS 2
Fires on annotation click	Property:annotationClick\$("#chart").ejChart({ annotationClick: () { } });	Not applicable
Fires after animation	Property:animationComplete\$("#chart").ejChart({ animationComplete: () { } });	Property:animationComplete()let chart: AccumulationChart = new AccumulationChart({ animationComplete: () => { } });chart.appendTo('#chart');
Fires on after chart resize	Property:afterResize\$("#chart").ejChart({ afterResize: () { } });	Not applicable
Fires on before chart resize	Property:beforeResize\$("#chart").ejChart({ beforeResize: () { } });	Property:resizedlet chart: AccumulationChart = new AccumulationChart({ resized: () => { } });chart.appendTo('#chart');
Fires on chart click	Property:chartClick\$("#chart").ejChart({ chartClick: () { } });	Property:chartMouseClickedlet chart: AccumulationChart = new AccumulationChart({

		<code>chartMouseClicked: () => { });chart.appendTo('#chart');</code>
Fires on chart mouse move	<code>Property:chartMouseMove\$("#chart").ejChart({ chartMouseMove: () { }});</code>	<code>Property:chartMouseMovelet chart: AccumulationChart = new AccumulationChart({ chartMouseMove: () => { });chart.appendTo('#chart');</code>
Fires on chart mouse leave	<code>Property:chartMouseLeave\$("#chart").ejChart({ chartMouseLeave: () { }});</code>	<code>Property:chartMouseLeavelet chart: AccumulationChart = new AccumulationChart({ chartMouseLeave: () => { });chart.appendTo('#chart');</code>
Fires on before chart double click	<code>Property:chartDoubleClick\$("#chart").ejChart({ chartDoubleClick: () { }});</code>	Not applicable
Fires on chart mouse up	Not Applicable	<code>Property:chartmouseUplet chart: AccumulationChart = new AccumulationChart({ chartmouseUp: () => { });chart.appendTo('#chart');</code>
Fires on chart mouse down	Not Applicable	<code>Property:chartmouseDownlet chart: AccumulationChart = new AccumulationChart({ chartmouseDown: () => { });chart.appendTo('#chart');</code>
Fires during the calculation of chart area bounds. You can use this event to customize the bounds of chart area	<code>Property:chartAreaBoundsCalculate\$("#chart").ejChart({ chartAreaBoundsCalculate: () { }});</code>	Not applicable
Fires when	<code>Property:destroy\$("#chart").ejChart({ destroy: () { }});</code>	Not applicable

chart is destroyed completely.		
Fires after chart is created.	Property:create\$("#chart").ejChart({ create: () { } });	Property:loadedlet chart: AccumulationChart = new AccumulationChart({ loaded: () => { } });chart.appendTo('#chart');
Fires before rendering the data labels.	Property:displayTextRendering\$("#chart").ejChart({ displayTextRendering: () { } });	Property:textRenderlet chart: AccumulationChart = new AccumulationChart({ textRender: () => { } });chart.appendTo('#chart');
Fires on clicking the legend item.	Property:legendItemClick\$("#chart").ejChart({ legendItemClick: () { } });	Not applicable
Fires when moving mouse over legend item	Property:legendItemMouseMove\$("#chart").ejChart({ legendItemMouseMove: () { } });	Not applicable
Fires before rendering the legend item.	Property:legendItemRendering\$("#chart").ejChart({ legendItemRendering: () { } });	Property:legendRenderlet chart: AccumulationChart = new AccumulationChart({ legendRender: () => { } });chart.appendTo('#chart');
Fires before loading the chart.	Property:load\$("#chart").ejChart({ load: () { } });	Property:loadlet chart: AccumulationChart = new AccumulationChart({ load: () => { } });chart.appendTo('#chart');
Fires on clicking a point in chart.	Property:pointRegionClick\$("#chart").ejChart({ pointRegionClick: () { } });	Property:pointClick let chart: AccumulationChart = new AccumulationChart({ pointClick : () => { } });chart.appendTo('#chart');

Fires when mouse is moved over a point.	<code>Property:pointRegionMouseMove\$("#chart").ejChart({ pointRegionMouseMove: () { }});</code>	<code>Property:pointMove let chart: AccumulationChart = new AccumulationChart({ pointMove : () => { }});chart.appendTo('#chart');</code>
Fires before rendering chart.	<code>Property:preRender\$("#chart").ejChart({ preRender: () { }});</code>	Not applicable
Fires when point render.	Not Applicable	<code>Property:pointRender let chart: AccumulationChart = new AccumulationChart({ pointRender : () => { }});chart.appendTo('#chart');</code>
Fires before rendering a series.	<code>Property:seriesRendering\$("#chart").ejChart({ seriesRendering: () { }});</code>	<code>Property:seriesRender let chart: AccumulationChart = new AccumulationChart({ seriesRender : () => { }});chart.appendTo('#chart');</code>
Fires before rendering the Chart title.	<code>Property:titleRendering\$("#chart").ejChart({ titleRendering: () { }});</code>	Not applicable
Fires before rendering the Chart sub title.	<code>Property:subTitleRendering\$("#chart").ejChart({ subTitleRendering: () { }});</code>	Not applicable
Fires before rendering the tooltip.	<code>Property:toolTipInitialize\$("#chart").ejChart({ toolTipInitialize: () { }});</code>	<code>Property:tooltipRender let chart: AccumulationChart = new AccumulationChart({ tooltipRender : () => { }});chart.appendTo('#chart');</code>

AppBar

Size and color in `##Platform_Name##` AppBar control

Size

The size of the AppBar can be set using the [mode](#) property. The available types of the AppBar are as follows:

- Regular AppBar
- Prominent AppBar
- Dense AppBar

Regular AppBar

This mode is the default one in which the AppBar is displayed with the default height.

INDEX.JS

```
var defaultAppBarObj = new ej.navigations.AppBar({
  colorMode: 'Primary'
});
defaultAppBarObj.appendTo("#defaultAppBar");
var defaultButtonMenuObj = new ej.buttons.Button({ cssClass: 'e-inherit',
iconCss: 'e-icons e-menu' });
defaultButtonMenuObj.appendTo('#defaultButtonMenu');
var defaultButtonLoginObj = new ej.buttons.Button({ cssClass: 'e-inherit',
content: 'FREE TRIAL' });
defaultButtonLoginObj.appendTo('#defaultButtonLogin');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 - AppBar</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.cs
s" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="control-section default-appbar-section">
      <div class="control-container">
        <header id="defaultAppBar">
          <button id="defaultButtonMenu"></button>
          <span class="regular">Regular AppBar</span>
          <div class="e-appbar-spacer"></div>
```

```

        <button id="defaultButtonLogin"></button>
      </header>
    </div>
  </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Prominent AppBar

This height mode can be set to the AppBar by setting **Prominent** to the property [mode](#). The prominent AppBar is displayed with a longer height and can be used for larger titles, images, or texts. It is also longer than the regular AppBar. In the following example, we have customized the prominent text using align-self and white-space CSS properties. You can change the prominent AppBar height if larger titles, images, or texts are used.

INDEX.JS

```

var defaultAppBarObj = new ej.navigations.AppBar({
  cssClass: 'prominent-appbar',
  colorMode: 'Primary',
  mode: 'Prominent'
});
defaultAppBarObj.appendTo("#defaultAppBar");
var defaultButtonMenuObj = new ej.buttons.Button({ cssClass: 'e-inherit',
  iconCss: 'e-icons e-menu' });
defaultButtonMenuObj.appendTo('#defaultButtonMenu');
var defaultButtonLoginObj = new ej.buttons.Button({ cssClass: 'e-inherit',
  content: 'FREE TRIAL' });
defaultButtonLoginObj.appendTo('#defaultButtonLogin');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 - AppBar</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
  user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link
href="https://cdn.jsdelivrivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.cs
s" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

```

```

<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="control-section default-appbar-section">
            <div class="control-container">
                <header id="defaultAppBar">
                    <button id="defaultButtonMenu"></button>
                    <span class="prominent">AppBar Component with Prominent
mode</span>

                    <div class="e-appbar-spacer"></div>
                    <button id="defaultButtonLogin"></button>
                </header>
            </div>
        </div>
    </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Dense AppBar

This height mode can be set to the AppBar by setting **Dense** to the property [mode](#). Dense AppBar is displayed with shorter height which is denser to accommodate all the AppBar content.

INDEX.JS

```

var defaultAppBarObj = new ej.navigations.AppBar({
    colorMode: 'Primary',
    mode: 'Dense'
});
defaultAppBarObj.appendTo("#defaultAppBar");
var defaultButtonMenuObj = new ej.buttons.Button({ cssClass: 'e-inherit',
iconCss: 'e-icons e-menu' });
defaultButtonMenuObj.appendTo('#defaultButtonMenu');
var defaultButtonLoginObj = new ej.buttons.Button({ cssClass: 'e-inherit',
content: 'FREE TRIAL' });
defaultButtonLoginObj.appendTo('#defaultButtonLogin');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 - AppBar</title>
    <meta charset="utf-8">

```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
<meta name="description" content="Essential JS 2">
<meta name="author" content="Syncfusion">
<link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.cs
s" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">
<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="control-section default-appbar-section">
            <div class="control-container">
                <header id="defaultAppBar">
                    <button id="defaultButtonMenu"></button>
                    <span class="dense">Dense AppBar</span>
                    <div class="e-appbar-spacer"></div>
                    <button id="defaultButtonLogin"></button>
                </header>
            </div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Color

The background and font colors can be set using the [colorMode](#) property. The available types of background color for the AppBar are as follows:

- Light AppBar
- Dark AppBar
- Primary AppBar
- Inherit AppBar

Light AppBar

This color mode is the default one in which the AppBar can be displayed with a light background and its corresponding font color.

INDEX.JS

```
var defaultAppBarObj = new ej.navigations.AppBar({});
defaultAppBarObj.appendTo("#defaultAppBar");
var defaultButtonLoginObj = new ej.buttons.Button({ isPrimary: true,
content: 'FREE TRIAL' });
defaultButtonLoginObj.appendTo('#defaultButtonLogin');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 - AppBar</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link
href="https://cdn.jsdelivrivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.cs
s" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="control-section default-appbar-section">
      <div class="control-container">
        <header id="defaultAppBar">
          <a href="https://www.syncfusion.com/javascript-ui-
controls" target="_blank" rel="noopener" role="link" aria-label="Syncfusion
javascript controls">
            <div class="syncfusion-logo"></div>
          </a>
          <div class="e-appbar-spacer"></div>
          <button id="defaultButtonLogin"></button>
        </header>
      </div>
    </div>
  </div>
</div>
```

```
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Dark AppBar

This color mode can be set to the AppBar by setting **Dark** to the property [colorMode](#). A dark AppBar can be displayed with a dark background and its corresponding font color.

INDEX.JS

```
var defaultAppBarObj = new ej.navigations.AppBar({
    colorMode: 'Dark'
});
defaultAppBarObj.appendTo("#defaultAppBar");
var defaultButtonMenuObj = new ej.buttons.Button({ cssClass: 'e-inherit',
iconCss: 'e-icons e-menu' });
defaultButtonMenuObj.appendTo('#defaultButtonMenu');
var defaultButtonLoginObj = new ej.buttons.Button({ cssClass: 'e-inherit',
content: 'FREE TRIAL' });
defaultButtonLoginObj.appendTo('#defaultButtonLogin');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 - AppBar</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Essential JS 2">
    <meta name="author" content="Syncfusion">
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.cs
s" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="styles.css" rel="stylesheet">
    <!--system js reference and configuration-->

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
```



```

<div id="container">
  <div class="control-section default-appbar-section">
    <div class="control-container">
      <header id="defaultAppBar">
        <button id="defaultButtonMenu"></button>
        <div class="e-appbar-spacer"></div>
        <button id="defaultButtonLogin"></button>
      </header>
    </div>
  </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Primary AppBar

This color mode can be set to the AppBar by setting **Primary** to the property [colorMode](#). The primary AppBar can be displayed with primary colors.

INDEX.JS

```

var defaultAppBarObj = new ej.navigations.AppBar({
  colorMode: 'Primary'
});
defaultAppBarObj.appendTo("#defaultAppBar");
var defaultButtonMenuObj = new ej.buttons.Button({ cssClass: 'e-inherit',
iconCss: 'e-icons e-menu' });
defaultButtonMenuObj.appendTo('#defaultButtonMenu');
var defaultButtonLoginObj = new ej.buttons.Button({ cssClass: 'e-inherit',
content: 'FREE TRIAL' });
defaultButtonLoginObj.appendTo('#defaultButtonLogin');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 - AppBar</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link
href="https://cdn.jsdelivrivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.cs
s" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">

```

```

<link href="styles.css" rel="stylesheet">
<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="control-section default-appbar-section">
            <div class="control-container">
                <header id="defaultAppBar">
                    <button id="defaultButtonMenu"></button>
                    <div class="e-appbar-spacer"></div>
                    <button id="defaultButtonLogin"></button>
                </header>
            </div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Inherit AppBar

This color mode can be set to the AppBar by setting **Inherit** to the property [colorMode](#). The AppBar inherits the background and font color from its parent element.

INDEX.JS

```

var defaultAppBarObj = new ej.navigations.AppBar({
    colorMode: 'Inherit'
});
defaultAppBarObj.appendTo("#defaultAppBar");
var defaultButtonLoginObj = new ej.buttons.Button({ isPrimary: true,
content: 'FREE TRIAL' });
defaultButtonLoginObj.appendTo('#defaultButtonLogin');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 - AppBar</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Essential JS 2">
    <meta name="author" content="Syncfusion">

```

```

<link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.cs
s" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="control-section default-appbar-section">
      <div class="control-container">
        <header id="defaultAppBar">
          <a href="https://www.syncfusion.com/javascript-ui-
controls" target="_blank" rel="noopener" role="link" aria-label="Syncfusion
javascript controls">
            <div class="syncfusion-logo"></div>
          </a>
          <div class="e-appbar-spacer"></div>
          <button id="defaultButtonLogin"></button>
        </header>
      </div>
    </div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Accessibility in ##Platform_Name## AppBar control

The AppBar control followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the AppBar control is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support | |

| [Section 508](#) Support | `` |

| Screen Reader Support | `` |

| Right-To-Left Support | `` |

| Color Contrast | `` |

| Mobile Device Support | `` |

| Keyboard Navigation Support | `` |

| [Accessibility Checker](#) Validation | `` |

| [Axe-core](#) Accessibility Validation | `` |

`<style>`

```
.post .post-content img {
display: inline-block;
margin: 0.5em 0;
}
```

`</style>`

`<div>` - All features of the control meet the requirement.`</div>`

`<div>` - Some features of the control do not meet the requirement.`</div>`

`<div>` - The control does not meet the requirement.`</div>`

Keyboard interaction

The AppBar control provides the focus element navigation based on its's tab key order.

Ensuring accessibility

The AppBar control accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the AppBar control is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the AppBar control with accessibility tools.

See also

- [Accessibility in Syncfusion components](#)

Position in ##Platform_Name## AppBar control

The position of the AppBar can be set using the [position](#) and [isSticky](#) property. The AppBar provides the following options for setting its position:

- Top AppBar
- Bottom AppBar
- Sticky AppBar

Top AppBar

The top AppBar is the default one in which it positions the AppBar at the top of the content.

INDEX.JS

```
var defaultAppBarObj = new ej.navigations.AppBar({
    colorMode: 'Primary'
});
defaultAppBarObj.appendTo("#defaultAppBar");
var defaultButtonMenuObj = new ej.buttons.Button({ cssClass: 'e-inherit',
    iconCss: 'e-icons e-menu' });
defaultButtonMenuObj.appendTo('#defaultButtonMenu');
var defaultButtonLoginObj = new ej.buttons.Button({ cssClass: 'e-inherit',
    content: 'FREE TRIAL' });
defaultButtonLoginObj.appendTo('#defaultButtonLogin');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 - AppBar</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link
href="https://cdn.jsdelivrivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.cs
s" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="control-section default-appbar-section">
```

```

<div class="control-container">
  <header id="defaultAppBar">
    <button id="defaultButtonMenu"></button>
    <div class="e-appbar-spacer"></div>
    <button id="defaultButtonLogin"></button>
  </header>
  <div class="appbar-content" style="font-size: 12px">
    <p>
      The AppBar also known as action bar or nav bar
      displays information and actions related to the current application screen.
      It is used to show branding, screen titles, navigation, and actions. The
      control supports height mode, color mode, positioning, and more.
    </p>
    <p>
      The AppBar control provides flexible ways to
      configure the look and feel of the bar to match your requirement.
    </p>
    <p>
      Developers can control the appearance and behaviors
      of the AppBar using a rich set of APIs.
    </p>
    <p>
      The AppBar component supports built-in themes such
      as Material, Bootstrap, Fabric (Office 365), Tailwind CSS, and high
      contrast. Users can customize these built-in themes or create new themes to
      achieve their desired look and feel by either simply overriding SASS
      variables or using our Theme Studio application.
    </p>
  </div>
</div>
</div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Bottom AppBar

This position can be set to the AppBar by setting **Bottom** to the property [position](#). The bottom AppBar positions the AppBar at the bottom of the content.

INDEX.JS

```

var defaultAppBarObj = new ej.navigations.AppBar({
  colorMode: 'Primary',
  position: 'Bottom'
});
defaultAppBarObj.appendTo("#defaultAppBar");
var defaultButtonMenuObj = new ej.buttons.Button({ cssClass: 'e-inherit',
iconCss: 'e-icons e-menu' });
defaultButtonMenuObj.appendTo('#defaultButtonMenu');

```

```
var defaultButtonLoginObj = new ej.buttons.Button({ cssClass: 'e-inherit',
content: 'FREE TRIAL' });
defaultButtonLoginObj.appendTo('#defaultButtonLogin');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 - AppBar</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.cs
s" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="control-section default-appbar-section">
      <div class="control-container">
        <header id="defaultAppBar">
          <button id="defaultButtonMenu"></button>
          <div class="e-appbar-spacer"></div>
          <button id="defaultButtonLogin"></button>
        </header>
        <div class="appbar-content" style="font-size: 12px">
          <p>
            The AppBar also known as action bar or nav bar
            displays information and actions related to the current application screen.
            It is used to show branding, screen titles, navigation, and actions. The
            control supports height mode, color mode, positioning, and more.
          </p>
          <p>
            The AppBar control provides flexible ways to
            configure the look and feel of the bar to match your requirement.
          </p>
          <p>
            Developers can control the appearance and behaviors
            of the AppBar using a rich set of APIs.
          </p>
        </div>
      </div>
    </div>
  </div>
```

```

        <p>
            The AppBar component supports built-in themes such
            as Material, Bootstrap, Fabric (Office 365), Tailwind CSS, and high
            contrast. Users can customize these built-in themes or create new themes to
            achieve their desired look and feel by either simply overriding SASS
            variables or using our Theme Studio application.
        </p>
    </div>
</div>
</div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Sticky AppBar

This position can be set to the AppBar by setting `true` to the property `isSticky`. AppBar will be sticky while scrolling the AppBar content.

INDEX.JS

```

var defaultAppBarObj = new ej.navigations.AppBar({
    colorMode: 'Primary',
    isSticky: true
});
defaultAppBarObj.appendTo("#defaultAppBar");
var defaultButtonMenuObj = new ej.buttons.Button({ cssClass: 'e-inherit',
iconCss: 'e-icons e-menu' });
defaultButtonMenuObj.appendTo('#defaultButtonMenu');
var defaultButtonLoginObj = new ej.buttons.Button({ cssClass: 'e-inherit',
content: 'FREE TRIAL' });
defaultButtonLoginObj.appendTo('#defaultButtonLogin');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 - AppBar</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Essential JS 2">
    <meta name="author" content="Syncfusion">
    <link
href="https://cdn.jsdelivrivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.cs
s" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">
<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="control-section default-appbar-section">
            <div class="control-container">
                <header id="defaultAppBar">
                    <button id="defaultButtonMenu"></button>
                    <div class="e-appbar-spacer"></div>
                    <button id="defaultButtonLogin"></button>
                </header>
                <div class="appbar-content" style="font-size: 12px">
                    <p>
                        The AppBar also known as action bar or nav bar
                        displays information and actions related to the current application screen.
                        It is used to show branding, screen titles, navigation, and actions. The
                        control supports height mode, color mode, positioning, and more.
                    </p>
                    <p>
                        The AppBar control provides flexible ways to
                        configure the look and feel of the bar to match your requirement.
                    </p>
                    <p>
                        Developers can control the appearance and behaviors
                        of the AppBar using a rich set of APIs.
                    </p>
                    <p>
                        The AppBar component supports built-in themes such
                        as Material, Bootstrap, Fabric (Office 365), Tailwind CSS, and high
                        contrast. Users can customize these built-in themes or create new themes to
                        achieve their desired look and feel by either simply overriding SASS
                        variables or using our Theme Studio application.
                    </p>
                </div>
            </div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Design in ##Platform_Name## AppBar control

Spacer

Spacer is used to provide spacing between the AppBar contents which gives additional space to the content layout.

The following example depicts the code to provide spacing between the home and pan buttons in the AppBar:

INDEX.JS

```
var defaultAppBarObj = new ej.navigations.AppBar({
    colorMode: 'Primary'
});
defaultAppBarObj.appendTo("#defaultAppBar");
var defaultButtonHomeObj = new ej.buttons.Button({ cssClass: 'e-inherit',
iconCss: 'e-icons e-home' });
defaultButtonHomeObj.appendTo('#defaultButtonHome');
var defaultButtonCutObj = new ej.buttons.Button({ cssClass: 'e-inherit',
iconCss: 'e-icons e-cut' });
defaultButtonCutObj.appendTo('#defaultButtonCut');
var defaultButtonPanObj = new ej.buttons.Button({ cssClass: 'e-inherit',
iconCss: 'e-icons e-pan' });
defaultButtonPanObj.appendTo('#defaultButtonPan');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 - AppBar</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Essential JS 2">
    <meta name="author" content="Syncfusion">
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.cs
s" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="control-section default-appbar-section">
            <div class="control-container">
                <header id="defaultAppBar">
```

```

        <button id="defaultButtonHome"></button>
        <div class="e-appbar-spacer"></div>
        <button id="defaultButtonCut"></button>
        <div class="e-appbar-spacer"></div>
        <button id="defaultButtonPan"></button>
    </header>
</div>
</div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Separator

Separator shows a vertical line to visually group or separate the AppBar contents.

The following example depicts the code to provide a vertical line between a group of buttons in the AppBar.

INDEX.JS

```

var defaultAppBarObj = new ej.navigations.AppBar({
    colorMode: 'Primary'
});
defaultAppBarObj.appendTo("#defaultAppBar");
var defaultButtonCutObj = new ej.buttons.Button({ cssClass: 'e-inherit',
iconCss: 'e-icons e-cut' });
defaultButtonCutObj.appendTo("#defaultButtonCut");
var defaultButtonCopyObj = new ej.buttons.Button({ cssClass: 'e-inherit',
iconCss: 'e-icons e-copy' });
defaultButtonCopyObj.appendTo("#defaultButtonCopy");
var defaultButtonPasteObj = new ej.buttons.Button({ cssClass: 'e-inherit',
iconCss: 'e-icons e-paste' });
defaultButtonPasteObj.appendTo("#defaultButtonPaste");
var defaultButtonBoldObj = new ej.buttons.Button({ cssClass: 'e-inherit',
iconCss: 'e-icons e-bold' });
defaultButtonBoldObj.appendTo("#defaultButtonBold");
var defaultButtonUnderlineObj = new ej.buttons.Button({ cssClass: 'e-
inherit', iconCss: 'e-icons e-underline' });
defaultButtonUnderlineObj.appendTo("#defaultButtonUnderline");
var defaultButtonItalicObj = new ej.buttons.Button({ cssClass: 'e-inherit',
iconCss: 'e-icons e-italic' });
defaultButtonItalicObj.appendTo("#defaultButtonItalic");
var defaultButtonAlignLeftObj = new ej.buttons.Button({ cssClass: 'e-
inherit', iconCss: 'e-icons e-align-left' });
defaultButtonAlignLeftObj.appendTo("#defaultButtonAlignLeft");
var defaultButtonAlignRightObj = new ej.buttons.Button({ cssClass: 'e-
inherit', iconCss: 'e-icons e-align-right' });
defaultButtonAlignRightObj.appendTo("#defaultButtonAlignRight");
var defaultButtonAlignCenterObj = new ej.buttons.Button({ cssClass: 'e-
inherit', iconCss: 'e-icons e-align-center' });

```

```
defaultButtonAlignCenterObj.appendTo('#defaultButtonAlignCenter');
var defaultButtonJustifyObj = new ej.buttons.Button({ cssClass: 'e-inherit',
iconCss: 'e-icons e-justify' });
defaultButtonJustifyObj.appendTo('#defaultButtonJustify');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 - AppBar</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.cs
s" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="control-section default-appbar-section">
      <div class="control-container">
        <header id="defaultAppBar">
          <button id="defaultButtonCut"></button>
          <button id="defaultButtonCopy"></button>
          <button id="defaultButtonPaste"></button>
          <div class="e-appbar-separator"></div>
          <button id="defaultButtonBold"></button>
          <button id="defaultButtonUnderline"></button>
          <button id="defaultButtonItalic"></button>
          <div class="e-appbar-separator"></div>
          <button id="defaultButtonAlignLeft"></button>
          <button id="defaultButtonAlignRight"></button>
          <button id="defaultButtonAlignCenter"></button>
          <button id="defaultButtonJustify"></button>
        </header>
      </div>
    </div>
  </div>
<script>
var ele = document.getElementById('container');
```

```

if(ele) {
    ele.style.visibility = "visible";
}

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Media Query

Media Query is used to adjusting the AppBar for different screen sizes. Resize the screen to observe the responsive layout of the AppBar.

INDEX.JS

```

var defaultAppBarObj = new ej.navigations.AppBar({
    colorMode: 'Primary'
});
defaultAppBarObj.appendTo("#defaultAppBar");
var defaultButtonMenuObj = new ej.buttons.Button({ cssClass: 'e-inherit',
iconCss: 'e-icons e-menu' });
defaultButtonMenuObj.appendTo('#defaultButtonMenu');
var defaultButtonHomeObj = new ej.buttons.Button({ cssClass: 'e-inherit',
content: 'Home' });
defaultButtonHomeObj.appendTo('#defaultButtonHome');
var defaultButtonAboutObj = new ej.buttons.Button({ cssClass: 'e-inherit',
content: 'About' });
defaultButtonAboutObj.appendTo('#defaultButtonAbout');
var defaultButtonProductsObj = new ej.buttons.Button({ cssClass: 'e-
inherit', content: 'Products' });
defaultButtonProductsObj.appendTo('#defaultButtonProducts');
var defaultButtonContactsObj = new ej.buttons.Button({ cssClass: 'e-
inherit', content: 'Contacts' });
defaultButtonContactsObj.appendTo('#defaultButtonContacts');
var defaultButtonLoginObj = new ej.buttons.Button({ cssClass: 'e-inherit',
content: 'Login' });
defaultButtonLoginObj.appendTo('#defaultButtonLogin');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 - AppBar</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Essential JS 2">
    <meta name="author" content="Syncfusion">
    <link
href="https://cdn.jsdelivrivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.cs
s" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="styles.css" rel="stylesheet">

```

```

<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="control-section default-appbar-section">
            <div class="control-container">
                <header id="defaultAppBar">
                    <button id="defaultButtonMenu"></button>
                    <button id="defaultButtonHome"></button>
                    <button id="defaultButtonAbout"></button>
                    <button id="defaultButtonProducts"></button>
                    <button id="defaultButtonContacts"></button>
                    <div class="e-appbar-spacer"></div>
                    <div class="e-appbar-separator"></div>
                    <button id="defaultButtonLogin"></button>
                </header>
            </div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Designing AppBar with Menu

AppBar is rendered with a Menu component in its AppBar header area. Menu component's styles are inherited from the AppBar component using the `e-inherit` CSS class.

INDEX.JS

```

var companyMenuItems = [
    {
        text : 'Company',
        items: [
            { text: 'About Us' },
            { text: 'Customers' },
            { text: 'Blog' },
            { text: 'Careers' }
        ]
    }
];
var productMenuItems = [
    {
        text : 'Products',
        items: [

```

```

        { text: 'Developer' },
        { text: 'Analytics' },
        { text: 'Reporting' },
        { text: 'Help Desk' }
    ]
}
];
var aboutMenuItems = [
    {
        text : 'About Us'
    }
];
var carrerMenuItems = [
    {
        text : 'Carrers'
    }
];
var defaultAppBarObj = new ej.navigations.AppBar({
    colorMode: 'Primary'
});
defaultAppBarObj.appendTo("#defaultAppBar");
var defaultButtonMenuObj = new ej.buttons.Button({ cssClass: 'e-inherit',
iconCss: 'e-icons e-menu' });
defaultButtonMenuObj.appendTo('#defaultButtonMenu');
var defaultMenuCompanyObj = new ej.navigations.Menu({ cssClass: 'e-inherit',
items: companyMenuItems });
defaultMenuCompanyObj.appendTo('#defaultMenuCompany');
var defaultMenuProductsObj = new ej.navigations.Menu({ cssClass: 'e-
inherit', items: productMenuItems });
defaultMenuProductsObj.appendTo('#defaultMenuProducts');
var defaultMenuAboutObj = new ej.navigations.Menu({ cssClass: 'e-inherit',
items: aboutMenuItems });
defaultMenuAboutObj.appendTo('#defaultMenuAbout');
var defaultMenuCarrersObj = new ej.navigations.Menu({ cssClass: 'e-inherit',
items: carrerMenuItems });
defaultMenuCarrersObj.appendTo('#defaultMenuCarrers');
var defaultButtonLoginObj = new ej.buttons.Button({ cssClass: 'e-inherit',
content: 'Login' });
defaultButtonLoginObj.appendTo('#defaultButtonLogin');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 - AppBar</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Essential JS 2">
    <meta name="author" content="Syncfusion">
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.cs
s" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">
<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="control-section default-appbar-section">
            <div class="control-container">
                <header id="defaultAppBar">
                    <button id="defaultButtonMenu"></button>
                    <ul id="defaultMenuCompany"></ul>
                    <ul id="defaultMenuProducts"></ul>
                    <ul id="defaultMenuAbout"></ul>
                    <ul id="defaultMenuCarrers"></ul>
                    <div class="e-appbar-spacer"></div>
                    <button id="defaultButtonLogin"></button>
                </header>
            </div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Designing AppBar with Buttons

The AppBar is rendered with a Button and DropDownButton component in its AppBar header area.

Button and DropDownButton components' styles are inherited from the AppBar component using the `e-inherit` CSS class.

INDEX.JS

```

var productDropDownButtonItem = [
    { text: 'Developer' },
    { text: 'Analytics' },
    { text: 'Reporting' },
    { text: 'E-Signature' },
    { text: 'Help Desk' }
];
var defaultAppBarObj = new ej.navigations.AppBar({
    colorMode: 'Primary'
});
defaultAppBarObj.appendTo("#defaultAppBar");

```



```

var defaultButtonMenuObj = new ej.buttons.Button({ cssClass: 'e-inherit',
iconCss: 'e-icons e-menu' });
defaultButtonMenuObj.appendTo('#defaultButtonMenu');
var defaultDropDownButtonProductObj = new ej.splitbuttons.DropDownButton({
cssClass: 'e-inherit', items: productDropDownButtonItem });
defaultDropDownButtonProductObj.appendTo('#defaultDropDownButtonProduct');
var defaultButtonLoginObj = new ej.buttons.Button({ cssClass: 'e-inherit',
content: 'Login' });
defaultButtonLoginObj.appendTo('#defaultButtonLogin');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 - AppBar</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link
href="https://cdn.jsdelivrivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.cs
s" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="control-section default-appbar-section">
      <div class="control-container">
        <header id="defaultAppBar">
          <button id="defaultButtonMenu"></button>
          <button
id="defaultDropDownButtonProduct">Products</button>
          <div class="e-appbar-spacer"></div>
          <button id="defaultButtonLogin"></button>
        </header>
      </div>
    </div>
  </div>
<script>
var ele = document.getElementById('container');

```

```

if(ele) {
    ele.style.visibility = "visible";
}

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Designing AppBar with SideBar

The AppBar is rendered with the SideBar component below the AppBar. Click on the menu icon to expand/collapse the Sidebar. In the following sample, the `toggle` method has been used to show or hide the Sidebar on the AppBar button click.

INDEX.JS

```

var data = [
    {
        nodeId: '01', nodeText: 'Installation',
    },
    {
        nodeId: '02', nodeText: 'Deployment',
    },
    {
        nodeId: '03', nodeText: 'Quick Start',
    },
    {
        nodeId: '04', nodeText: 'Components',
        nodeChild: [
            { nodeId: '04-01', nodeText: 'Calendar' },
            { nodeId: '04-02', nodeText: 'DatePicker' },
            { nodeId: '04-03', nodeText: 'DateTimePicker' },
            { nodeId: '04-04', nodeText: 'DateRangePicker' },
            { nodeId: '04-05', nodeText: 'TimePicker' },
            { nodeId: '04-06', nodeText: 'SideBar' }
        ]
    }
];

var defaultAppBarObj = new ej.navigations.AppBar({});
defaultAppBarObj.appendTo("#defaultAppBar");
var defaultButtonMenuObj = new ej.buttons.Button({ cssClass: 'e-inherit',
iconCss: 'e-icons e-menu' });
defaultButtonMenuObj.appendTo('#defaultButtonMenu');
defaultButtonMenuObj.element.addEventListener("click", toggle);
var sidebarMenu = new ej.navigations.Sidebar({
    width: '220px',
    target: '.main-content',
    mediaQuery: '(min-width: 600px)',
    isOpen: true
});
sidebarMenu.appendTo('#sideTree');
var inputObj = new ej.inputs.TextBox({
    placeholder: "Search..."
});
inputObj.appendTo("#resSearch");
var mainTreeView = new ej.navigations.TreeView({
    cssClass: "main-treeview",

```

```

    fields: { dataSource: data, id: 'nodeId', text: 'nodeText', child:
'nodeChild', iconCss: "iconCss" },
    expandOn: 'Click'
});
mainTreeView.appendTo('#mainTree');
function toggle() {
    sidebarMenu.toggle();
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 - AppBar</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link
href="https://cdn.jsdelivrivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.cs
s" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="wrapper" class="control-section default-appbar-section">
      <div class="control-container">
        <div>
          <header id="defaultAppBar">
            <button id="defaultButtonMenu"></button>
            <div class="e-folder">
              <div class="e-folder-name">Navigation Pane</div>
            </div>
          </header>
        </div>
        <aside id="sideTree" class="sidebar-treeview">
          <div class="main-menu">
            <div class="table-content">
              <input id="resSearch">

```

```

    <p class="main-menu-header">TABLE OF
CONTENTS</p>
    </div>
  </div>
  <div id="mainTree"></div>
</div>
</aside>
<div class="main-content" id="main-text">
  <div class="sidebar-content">
    <div class="sidebar-heading"> <h4>Responsive Sidebar
with Treeview</h4></div>
    <p class="paragraph-content">
      This is a graphical aid for visualising and
categorising the site, in the style of an expandable and
collapsable treeview component.
      It auto-expands to display the node(s), if any,
corresponding to the currently viewed title,
      highlighting that node(s)
      and its ancestors. Load-on-demand when expanding
nodes is available where supported (most graphical
browsers),
      falling back to a full-page reload. MediaWiki-
supported caching, aside from squid, has been considered
so that
      unnecessary re-downloads of content are avoided
where possible. The complete expanded/collapsed state of
      the treeview persists across page views in most
situations.
    </p>
  </div>
</div>
</div>
</div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Style and appearance in ##Platform_Name## AppBar control

To modify the AppBar appearance, you need to override the default CSS of the AppBar component. Please find the list of CSS classes and their corresponding sections in the AppBar component. Also, you have an option to create your own custom theme for the controls using our [Theme Studio](#).

CSS Class	Purpose of Class
text-align: center;	Aligns text horizontally in the center of the container.
text-align: left;	Aligns text horizontally to the left of the container.
text-align: right;	Aligns text horizontally to the right of the container.
text-align: justify;	Aligns text horizontally to justify the lines (stretches or shrinks words to fill the line).
text-align: left; text-align: right;	Combines left and right alignment (e.g., for a signature block).
text-align: center; text-align: left;	Combines center and left alignment (e.g., for a centered heading with left-aligned text below it).
text-align: center; text-align: right;	Combines center and right alignment (e.g., for a centered heading with right-aligned text below it).
text-align: center; text-align: justify;	Combines center and justify alignment (e.g., for a centered heading with justified text below it).
text-align: center; text-align: left; text-align: right;	Combines center, left, and right alignment (e.g., for a centered heading with left-aligned text on the left and right-aligned text on the right).
text-align: center; text-align: justify; text-align: left;	Combines center, justify, and left alignment (e.g., for a centered heading with justified text in the middle and left-aligned text on the left).
text-align: center; text-align: justify; text-align: right;	Combines center, justify, and right alignment (e.g., for a centered heading with justified text in the middle and right-aligned text on the right).
text-align: center; text-align: justify; text-align: left; text-align: right;	Combines center, justify, left, and right alignment (e.g., for a centered heading with justified text in the middle and left-aligned text on the left and right-aligned text on the right).

— — — — —

`|.e-appbar|` To customize the appbar.

|.e-appbar.e-prominent|To customize the prominent appbar.

|.e-appbar.e-dense|To customize the dense appbar.|

|.e-appbar.e-light|To customize the light appbar.|

|.e-appbar.e-dark|To customize the dark appbar.|

|.e-appbar.e-primary|To customize the dark appbar.|

|.e-appbar.e-inherit|To customize the inherit appbar.|

Note: You can change the prominent AppBar height if larger titles, images, or texts are used.

CssClass

CssClass is used for AppBar customization based on the custom class. In the example below, the AppBar background and color are customized using the [cssClass](#) property.

INDEX.JS

```
var defaultAppBarObj = new ej.navigations.AppBar({
  colorMode: 'Primary',
  cssClass: 'custom-appbar'
});
defaultAppBarObj.appendTo("#defaultAppBar");
var defaultButtonMenuObj = new ej.buttons.Button({ cssClass: 'e-inherit',
  iconCss: 'e-icons e-home' });
defaultButtonMenuObj.appendTo('#defaultButtonMenu');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 - AppBar</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link
href="https://cdn.jsdelivrivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.cs
s" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="control-section default-appbar-section">
```

```

        <div class="control-container">
            <header id="defaultAppBar">
                <button id="defaultButtonMenu"></button>
            </header>
        </div>
    </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

HtmlAttributes

It can be used for additional inline attributes by specifying as inline attributes or by specifying htmlAttributes directive. In the code example below, the aria-label of the AppBar is customized by specifying as attributes.

INDEX.JS

```

var defaultAppBarObj = new ej.navigations.AppBar({
    colorMode: 'Primary'
});
defaultAppBarObj.appendTo("#defaultAppBar");
var defaultButtonMenuObj = new ej.buttons.Button({ cssClass: 'e-inherit',
iconCss: 'e-icons e-home' });
defaultButtonMenuObj.appendTo('#defaultButtonMenu');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 - AppBar</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Essential JS 2">
    <meta name="author" content="Syncfusion">
    <link
href="https://cdn.jsdelivrivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.cs
s" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <!--system js reference and configuration-->

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>

```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="control-section default-appbar-section">
            <div class="control-container">
                <header id="defaultAppBar" aria-label="appbar">
                    <button id="defaultButtonMenu"></button>
                </header>
            </div>
        </div>
    </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

AutoComplete

Data binding in ##Platform_Name## Auto complete control

The AutoComplete loads the data either from local data sources or remote data services using the [dataSource](#) property. It supports the data type of array or **DataManager**.

The AutoComplete also supports different kind of data services such as OData, OData V4, Web API and data formats such as XML, JSON, JSONP with the help of DataManager Adaptors.

Fields	Type	Description
value	number or string	Specifies the hidden data value mapped to each list item that should contain a unique value.
groupBy	string	Specifies the category under which the list item has to be grouped.
iconCss	string	Specifies the icon class of each list item.

While binding complex data to AutoComplete, fields should be mapped correctly. Otherwise, the selected

item remains undefined.

Bind to local data

Local data can be represented in two ways as described below.

Array of string

The AutoComplete has support to load array of primitive data such as strings and numbers.

INDEX.JS

```

var sportsData = ['Badminton', 'Cricket', 'Football', 'Golf', 'Tennis'];

```

```
// initialize AutoComplete component
var listObj = new ej.dropdowns.AutoComplete({
  dataSource: sportsData,
  // set placeholder to AutoComplete input element
  placeholder: "Select games"});
listObj.appendTo('#atcelement');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 AutoComplete</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:0 auto; width:250px;">
    <input type="text" tabindex="1" id="atcelement">
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Array of object

The AutoComplete can generate its list items through an array of complex data. For this, the appropriate columns should be mapped to the [fields](#) property.

In the following example, **Game** column from complex data have been mapped to the **value** field.

INDEX.JS

```
let sportsData = [
  { Id: 'Game1', Game: 'Badminton' },
  { Id: 'Game2', Game: 'Basketball' },
  { Id: 'Game3', Game: 'Cricket' },
```



```

    { Id: 'Game4', Game: 'Football' },
    { Id: 'Game5', Game: 'Golf' },
    { Id: 'Game6', Game: 'Hockey' },
    { Id: 'Game7', Game: 'Rugby' },
    { Id: 'Game8', Game: 'Snooker' }
  ];

  // initialize AutoComplete component
  var listObj = new ej.dropdowns.AutoComplete({
    //set the data to dataSource property
    dataSource: sportsData,
    // maps the appropriate column to fields property
    fields: { value: 'Game' },
    // set placeholder to AutoComplete input element
    placeholder: "Find a game"});
  listObj.appendTo('#atcelement');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 AutoComplete</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:0 auto; width:250px;">
    <br>
    <input type="text" id="atcelement">
  </div>
  <script>
  var ele = document.getElementById('container');
  if(ele) {
    ele.style.visibility = "visible";
  }

  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Array of complex object

The AutoComplete can generate its list items through an array of complex data. For this, the appropriate columns should be mapped to the [fields](#) property.

In the following example, `Country.Name` column from complex data have been mapped to the `value` field.

INDEX.JS

```
let countriesData = [
    { Country: { Name: 'Australia' }, Code: { Id: 'AU' } },
    { Country: { Name: 'Bermuda' }, Code: { Id: 'BM' } },
    { Country: { Name: 'Canada' }, Code: { Id: 'CA' } },
    { Country: { Name: 'Cameroon' }, Code: { Id: 'CM' } },
    { Country: { Name: 'Denmark' }, Code: { Id: 'DK' } },
    { Country: { Name: 'France' }, Code: { Id: 'FR' } },
    { Country: { Name: 'Finland' }, Code: { Id: 'FI' } },
    { Country: { Name: 'Germany' }, Code: { Id: 'DE' } },
    { Country: { Name: 'Greenland' }, Code: { Id: 'GL' } },
    { Country: { Name: 'Hong Kong' }, Code: { Id: 'HK' } },
    { Country: { Name: 'India' }, Code: { Id: 'IN' } },
    { Country: { Name: 'Italy' }, Code: { Id: 'IT' } },
    { Country: { Name: 'Japan' }, Code: { Id: 'JP' } },
    { Country: { Name: 'Mexico' }, Code: { Id: 'MX' } },
    { Country: { Name: 'Norway' }, Code: { Id: 'NO' } },
    { Country: { Name: 'Poland' }, Code: { Id: 'PL' } },
    { Country: { Name: 'Switzerland' }, Code: { Id: 'CH' } },
    { Country: { Name: 'United Kingdom' }, Code: { Id: 'GB' } },
    { Country: { Name: 'United States' }, Code: { Id: 'US' } }
];

// initialize AutoComplete component
var listObj = new ej.dropdowns.AutoComplete({
    //set the data to dataSource property
    dataSource: countriesData,
    // maps the appropriate column to fields property
    fields: { value: 'Country.Name' },
    // set placeholder to AutoComplete input element
    placeholder: "Find a country"});
listObj.appendTo('#atcelement');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 AutoComplete</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin:0 auto; width:250px;">
        <br>
        <input type="text" id="atcelement">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Bind to remote data

The AutoComplete supports retrieval of data from remote data services with the help of **DataManager** component. The **Query** property is used to fetch data from the database and bind it to the AutoComplete.

The following sample displays the first 6 contacts from the **Customers** table of the **Northwind** data service.

INDEX.JS

```

//initiates the component
var acObject = new ej.dropdowns.AutoComplete({
    //bind the data manager instance to dataSource property
    dataSource: new ej.data.DataManager({
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/',
        adaptor: new ej.data.ODataV4Adaptor(),
        crossDomain: true
    }),
    //bind the Query instance to query property
    query: new ej.data.Query().from('Employees').select(['FirstName',
'City', 'EmployeeID']).take(6),
    //map the appropriate columns to fields property
    fields: { value: 'FirstName' },
    //set the placeholder to AutoComplete input
    placeholder: "Select an employee",
    //sort the resulted items
    sortOrder: 'Ascending'
});
//render the component
acObject.appendTo('#atcelement');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>

```

```

<title>Essential JS 2 AutoComplete</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="styles.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin:0 auto; width:250px;">
        <br>
        <input type="text" id="atcelement">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [How to load data using template](#)
- [How to group the data using header](#)
- [How to filter the bound data](#)

Templates in ##Platform_Name## Auto complete control

The AutoComplete has been provided with several options to customize each list items, group title, header and footer elements. It uses the Essential JS 2 [Template engine](#) to compile and render the elements properly.

Item template

The content of each list item within the AutoComplete can be customized with the help of [itemTemplate](#) property.

In the following sample, each list item is split into two columns to display relevant data's.

INDEX.JS

```
//initiates the component
```

```

var acObject = new ej.dropdowns.AutoComplete({
    //bind the data manager instance to dataSource property
    dataSource: new ej.data.DataManager({
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/',
        adaptor: new ej.data.ODataV4Adaptor(),
        crossDomain: true
    }),
    //bind the Query instance to query property
    query: new ej.data.Query().from('Employees').select(['FirstName',
'City', 'EmployeeID']).take(6),
    //map the appropriate columns to fields property
    fields: { value: 'FirstName' },
    //set the placeholder to AutoComplete input
    placeholder: "Select an employee",
    //sort the resulted items
    sortOrder: 'Ascending',
    //set the value to itemTemplate property
    itemTemplate: "<span><span class='name'>${FirstName}</span><span class
='city'>${City}</span></span>"
});
//render the component
acObject.appendTo('#atcelement');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 AutoComplete</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin:0 auto; width:250px;">
        <input type="text" id="atcelement">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>

```

```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Group template

The group header title under which appropriate sub-items are categorized can also be customize with the help of [groupTemplate](#) property. This template is common for both inline and floating group header template.

In the following sample, employees are grouped according to their city.

INDEX.JS

```
var groupPredicate = new ej.data.Predicate('City',
'equal','london').or('City','equal','seattle');
//initiates the component
var acObject = new ej.dropdowns.AutoComplete({
  //bind the data manager instance to dataSource property
  dataSource: new ej.data.DataManager({
    url: 'https://services.odata.org/V4/Northwind/Northwind.svc/',
    adaptor: new ej.data.ODataV4Adaptor(),
    crossDomain: true
  }),
  //bind the Query instance to query property
  query: new ej.data.Query().from('Employees').select(['FirstName',
'City','EmployeeID']).take(6).where(groupPredicate),
  //map the appropriate columns to fields property
  fields: { value: 'FirstName', groupBy: 'City'},
  //set the placeholder to AutoComplete input
  placeholder:"Select an employee",
  //sort the resulted items
  sortOrder: 'Ascending',
  //set the value to groupTemplate
  groupTemplate: "<strong>${City}</strong>"
});
//render the component
acObject.appendTo('#atcelement');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 AutoComplete</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin:0 auto; width:250px;">
        <input type="text" id="atcelement">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Header template

The header element is shown statically at the top of the suggestion list items within the AutoComplete, and any custom element can be placed as a header element using [headerTemplate](#) property.

In the following sample, the list items and its headers are designed and displayed as two columns similar to multiple columns of the grid.

INDEX.JS

```

//initiates the component
var acObject = new ej.dropdowns.AutoComplete({
    //bind the data manager instance to dataSource property
    dataSource: new ej.data.DataManager({
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/',
        adaptor: new ej.data.ODataV4Adaptor(),
        crossDomain: true
    }),
    //bind the Query instance to query property
    query: new ej.data.Query().from('Employees').select(['FirstName',
'City', 'EmployeeID']).take(6),
    //map the appropriate columns to fields property
    fields: { value: 'FirstName'},
    //set the placeholder to AutoComplete input
    placeholder: "Select an employee",
    //sort the resulted items
    sortOrder: 'Ascending',
    //set the value to headerTemplate
    headerTemplate: "<span class='head'><span class='name'>Name</span><span
class='city'>City</span></span>",
    //set the value to item template
    itemTemplate: "<span class='item' ><span
class='name'>${FirstName}</span><span class='city'>${City}</span></span>"
});
//render the component
acObject.appendTo('#atcelement');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 AutoComplete</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:0 auto; width:250px;">
    <input type="text" id="atcelement">
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Footer template

The AutoComplete has options to show a footer element at the bottom of the list items in the suggestion list. Here, you can place any custom element as a footer element using [footerTemplate](#) property.

In the following sample, footer element displays the total number of list items present in the AutoComplete.

INDEX.JS

```

var sportsData = [ "Basketball", "Cricket", "Football", "Golf"];
//initiates the component
var acObject = new ej.dropdowns.AutoComplete({
  //bind the data manager instance to dataSource property
  dataSource: sportsData ,
  //set the placeholder to AutoComplete input
  placeholder:"Select games",
  //set the value to footer template
  footerTemplate:"<span class='foot'> Total list items: "+
sportsData.length +"</span>"

```



```
});

//render the component
acObject.appendTo('#atcelement');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 AutoComplete</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:0 auto; width:250px;">

    <input type="text" id="atcelement">
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

No records template

The AutoComplete is provided with support to custom design the suggestion list content when no data is found and no matches found on search with the help of [noRecordsTemplate](#) property.

In the following sample, suggestion list content displays the notification of no data available.

INDEX.JS

```
//initiates the component
var acObject = new ej.dropdowns.AutoComplete({
  //bind the data manager instance to dataSource property
  dataSource: [],
  //set the placeholder to AutoComplete input
  placeholder:"Select an item",
```

```
//set the value to noRecords template
noRecordsTemplate:"<span class='norecord'> NO DATA AVAILABLE</span>"
});
//render the component
acObject.appendTo('#atcelement');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 AutoComplete</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:0 auto; width:250px;">
    <input type="text" tabindex="1" id="atcelement">
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Action failure template

There is also an option to custom design the suggestion list content when the data fetch request fails at the remote server. This can be achieved using the [actionFailureTemplate](#) property.

In the following sample, when the data fetch request fails, the AutoComplete displays the notification.

INDEX.JS

```
//initiates the component
var acObject = new ej.dropdowns.AutoComplete({
  //bind the data manager instance to dataSource property
  dataSource: new ej.data.DataManager({
    // Here, use the wrong url to display the action failure
    template
```

```

        url: 'https://services.odata.org/V4/Northwind/Northwind.svcs/',
        adaptor: new ej.data.ODataV4Adaptor(),
        crossDomain: true
    )),
    //bind the Query instance to query property
    query: new
ej.data.Query().from('Employees').select(['FirstName']).take(6),
    //map the appropriate columns to fields property
    fields: { text: 'FirstName', value: 'EmployeeID' },
    //set the placeholder to AutoComplete input
    placeholder: "Select an employee",
    //set the value to action failure template
    actionFailureTemplate: "<span class='action-failure'> Data fetch get
fails</span>"
});
acObject.appendTo('#atcelement');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 AutoComplete</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin:0 auto; width:250px;">
        <input type="text" tabindex="1" id="atcelement">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [How to achieve filtering](#)
- [How to group the data using header](#)
- [How to show the list items with icon](#)

Grouping in ##Platform_Name## Auto complete control

The AutoComplete supports wrapping nested elements into a group based on different categories. The category of each list item can be mapped through the [groupBy](#) field in the data table. The group header is displayed as both inline and fixed headers. The fixed group header content is updated dynamically on scrolling the suggestion list with its category value.

In the following sample, vegetables are grouped according on its category using `groupBy` field.

INDEX.JS

```
var vegetableData = [
  { Vegetable: 'Cabbage', Category: 'Leafy and Salad', Id: 'item1' },
  { Vegetable: 'Spinach', Category: 'Leafy and Salad', Id: 'item2' },
  { Vegetable: 'Wheat grass', Category: 'Leafy and Salad', Id: 'item3' },
  { Vegetable: 'Yarrow', Category: 'Leafy and Salad', Id: 'item4' },
  { Vegetable: 'Pumpkins', Category: 'Leafy and Salad', Id: 'item5' },
  { Vegetable: 'Chickpea', Category: 'Beans', Id: 'item6' },
  { Vegetable: 'Green bean', Category: 'Beans', Id: 'item7' },
  { Vegetable: 'Horse gram', Category: 'Beans', Id: 'item8' },
  { Vegetable: 'Garlic', Category: 'Bulb and Stem', Id: 'item9' },
  { Vegetable: 'Nopal', Category: 'Bulb and Stem', Id: 'item10' },
  { Vegetable: 'Onion', Category: 'Bulb and Stem', Id: 'item11' }
];
//initiate the AutoComplete
var vegetables = new ej.dropdowns.AutoComplete({
  //set the grouped data to dataSource property
  dataSource: vegetableData,
  // map the groupBy field with Category column
  fields: { groupBy: 'Category', value: 'Vegetable' },
  // set the placeholder to the AutoComplete input
  placeholder: "Find a vegetable"
});
//render the AutoComplete component
vegetables.appendTo('#atcelement');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 AutoComplete</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
```

```
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin:0 auto; width:250px;">
        <br>
        <input type="text" id="atcelement">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Customization

The grouping header is also provided with customization option. This allows custom designing using the [groupTemplate](#) property for both inline and fixed headers as referred here:

See Also

- [Group Template support to AutoComplete.](#)

Filtering in ##Platform_Name## Auto complete control

The AutoComplete has built-in support to filter data items. The filter operation starts as soon as you start typing characters in the component.

Change the filter type

Determines on which filter type, the component needs to be considered on search action. The available [filterType](#) and its supported data types are

Filter Type	Description	Supported Types
---	---	---
StartsWith	Checks whether a value begins with the specified value.	String
EndsWith	Checks whether a value ends with specified value.	String
Contains	Checks whether a value contains with specified value.	String

The following examples shows the data filtering is done with **StartsWith** type.

INDEX.JS

```
//initiates the component
var acObject = new ej.dropdowns.AutoComplete({
```

```

//bind the data manager instance to dataSource property
dataSource: new ej.data.DataManager({
    url:
'https://services.odata.org/V4/Northwind/Northwind.svc/Customers',
    adaptor: new ej.data.ODataV4Adaptor(),
    crossDomain: true
}),
//bind the Query instance to query property
query: new ej.data.Query().select(['ContactName']),
//map the appropriate columns to fields property
fields: { value: 'ContactName'},
//set the placeholder to AutoComplete input
placeholder: "Find a customer",
//set the filterType to searching operation
filterType: 'StartsWith',
//sort the resulted items
sortOrder: 'Ascending'
});
//render the component
acObject.appendTo('#atcelement');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 AutoComplete</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:0 auto; width:250px;">
    <br>
    <input type="text" id="atcelement">
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
<script src="index.js" type="text/javascript"></script>

```

```
</body></html>
```

Filter item count

You can specify the filter suggestion item count through [suggestionCount](#) property of AutoComplete.

The following example, to restrict the suggestion list item counts as 5.

INDEX.JS

```
//initiates the component
var acObject = new ej.dropdowns.AutoComplete({
  //bind the data manager instance to dataSource property
  dataSource: new ej.data.DataManager({
    url:
    'https://services.odata.org/V4/Northwind/Northwind.svc/Customers',
    adaptor: new ej.data.ODataV4Adaptor(),
    crossDomain: true
  }),
  //bind the Query instance to query property
  query: new ej.data.Query().select(['ContactName']),
  //map the appropriate columns to fields property
  fields: { value: 'ContactName' },
  //set the suggestionCount to show the maximum suggestion list item
  suggestionCount: 5,
  //set the placeholder to AutoComplete input
  placeholder: "Find a customer",
  //set the filterType to searching operation
  filterType: 'StartsWith',
  //sort the resulted items
  sortOrder: 'Ascending'
});
//render the component
acObject.appendTo('#atcelement');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 AutoComplete</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
```

```
<body>

  <div id="container" style="margin:0 auto; width:250px;">
    <br>
    <input type="text" id="atcelement">
  </div>
</body>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Limit the minimum filter character

You can set the limit for the character count to filter the data on the AutoComplete. This can be done by set the [minLength](#) property to AutoComplete.

In the following example, the remote request doesn't fetch the search data, until the search key contains three characters.

INDEX.JS

```
//initiates the component
var acObject = new ej.dropdowns.AutoComplete({
  //bind the data manager instance to dataSource property
  dataSource: new ej.data.DataManager({
    url:
    'https://services.odata.org/V4/Northwind/Northwind.svc/Customers',
    adaptor: new ej.data.ODataV4Adaptor(),
    crossDomain: true
  }),
  //bind the Query instance to query property
  query: new ej.data.Query().select(['ContactName']),
  //map the appropriate columns to fields property
  fields: { value: 'ContactName' },
  //set the minLength to restrict the remote request until search key
  //contains 3 characters.
  minLength: 3,
  //set the placeholder to AutoComplete input
  placeholder: "Find a customer",
  //set the filterType to searching operation
  filterType: 'StartsWith',
  //sort the resulted items
  sortOrder: 'Ascending'
});
//render the component
acObject.appendTo('#atcelement');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 AutoComplete</title>
  <meta charset="utf-8">
```



```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="styles.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin:0 auto; width:250px;">
        <br>
        <input type="text" id="atcelement">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Case sensitive filtering

Data items can be filtered either with or without case sensitivity using the DataManager. This can be done by setting the [ignoreCase](#) property of AutoComplete.

The following sample depicts how to filter the data with case-sensitive.

INDEX.JS

```

//initiates the component
var acObject = new ej.dropdowns.AutoComplete({
    //bind the data manager instance to dataSource property
    dataSource: new ej.data.DataManager({
        url:
        'https://services.odata.org/V4/Northwind/Northwind.svc/Customers',
        adaptor: new ej.data.ODataV4Adaptor(),
        crossDomain: true
    }),
    //bind the Query instance to query property
    query: new ej.data.Query().select(['ContactName']),
    //map the appropriate columns to fields property
    fields: { value: 'ContactName' },
    //set the ignoreCase as false to enable the case sensitive filtering
    ignoreCase: false,
    //set the placeholder to AutoComplete input

```

```
placeholder: "Find a customer",
//set the filterType to searching operation
filterType: 'StartsWith',
//sort the resulted items
sortOrder: 'Ascending'
});
//render the component
acObject.appendTo('#atcelement');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 AutoComplete</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:0 auto; width:250px;">
    <br>
    <input type="text" id="atcelement">
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Diacritics Filtering

An AutoComplete supports diacritics filtering which will ignore the [diacritics](#) and makes it easier to filter the results in international characters lists when the [ignoreAccent](#) is enabled.

In the following sample, data with diacritics are bound as dataSource for AutoComplete.

INDEX.JS

```
// create local data
var data = [
```

```

        'Aeróbics',
        'Aeróbics en Agua',
        'Aerografía',
        'Aerodelaje',
        'Águilas',
        'Ajedrez',
        'Ala Delta',
        'Álbumes de Música',
        'Alusivos',
        'Análisis de Escritura a Mano'];
// initialize AutoComplete component
var atcObj = new ej.dropdowns.AutoComplete({
    //set the local data to dataSource property
    dataSource: data,
    // set the placeholder to AutoComplete input element
    placeholder: 'e.g: aero',
    // enabled the ignoreAccent property for ignore the diacritics
    ignoreAccent: true
});
atcObj.appendTo('#atcelement');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 AutoComplete</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:0 auto; width:250px;">
    <br>
    <input type="text" id="atcelement">
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
<script src="index.js" type="text/javascript"></script>

```

```
</body></html>
```

See Also

- [How to achieve autofill while filtering](#)
- [How to group the data using header](#)
- [How to highlight the search data](#)

Virtualization in AutoComplete Component

AutoComplete virtualization is a technique used to efficiently render extensive lists of items while minimizing the impact on performance. This method is particularly advantageous when dealing with large datasets because it ensures that only a fixed number of DOM (Document Object Model) elements are created. When scrolling through the list, existing DOM elements are reused to display relevant data instead of generating new elements for each item. This recycling process is managed internally.

During virtual scrolling, the data retrieved from the data source depends on the popup height and the calculation of the list item height. Enabling the [enableVirtualization](#) option in a AutoComplete activates this virtualization technique.

When fetching data from the data source, the [actionBegin](#) event is triggered before data retrieval begins. Then, the [actionComplete](#) event is triggered once the data is successfully fetched.

When the enableVirtualization property is enabled, the `skip` and `take` properties provided by the user within the Query class at the initial state or during the `actionBegin` or `actionComplete` events will not be considered, since it is internally managed and calculated based on certain dimensions with respect to the popup height.

Binding local data

The AutoComplete can generate its list items through an array of complex data. For this, the appropriate columns should be mapped to the [fields](#) property. When using virtual scrolling, the list updates based on the scroll offset value, triggering a request to fetch more data from the server. As the data is being fetched, the `actionBegin` event occurs before the data retrieval starts. Once the data retrieval is successful, the `actionComplete` event is triggered, indicating that the data fetch process is complete.

In the following example, `text` column from complex data have been mapped to the `value` field.

INDEX.JS

```
var records = [];
for (var i = 1; i <= 150; i++) {
    var item = {
        id: 'id' + i,
        text: "Item " + i,
    };
    records.push(item);
}
//initiates the component
var acObject = new ej.dropdowns.AutoComplete({
    //bind the data source property
    dataSource: records,
    //map the appropriate columns to fields property
    fields: { value: 'text' },
    //set the placeholder to DropDownList input
```

```
placeholder:"Select an Item ",
//set enableVirtualization property to true
enableVirtualization: true,
//set the height of the popup element
popupHeight: '200px'
});
//render the component
acObject.appendTo('#atcelement');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 AutoComplete</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:0 auto; width:250px;">
    <input type="text" id="atcelement">
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Binding Remote data

The AutoComplete supports retrieval of data from remote data services with the help of **DataManager** component. When using remote data, it initially fetches all the data from the server, triggering the **actionBegin** and **actionComplete** events, and then stores the data locally. During virtual scrolling, additional data is retrieved from the locally stored data, triggering the **actionBegin** and **actionComplete** events at that time as well.

The following sample displays the OrderId from the **Orders** Data Service.

INDEX.JS

```
//initiates the component
var acObject = new ej.dropdowns.AutoComplete({
  //bind the dataSource property
  dataSource: new ej.data.DataManager({
    url: 'https://ej2services.syncfusion.com/js/development/api/orders',
    adaptor: new ej.data.WebApiAdaptor(),
    crossDomain: true
  }),
  //map the appropriate columns to fields property
  fields: { value: 'OrderID' },
  //set the placeholder to DropDownList input
  placeholder: "Select an ID ",
  //set enableVirtualization property to true
  enableVirtualization: true,
  //set the height of the popup element
  popupHeight: '200px'
});
//render the component
acObject.appendTo('#atcelement');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 AutoComplete</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:0 auto; width:250px;">
    <input type="text" id="atcelement">
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
```

```

}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Grouping with Virtualization

The AutoComplete component supports grouping with Virtualization. It allows you to organize elements into groups based on different categories. Each item in the list can be classified using the [groupBy](#) field in the data table. After grouping, virtualization works similarly to local data binding, providing a seamless user experience. When the data source is bound to remote data, an initial request is made to retrieve all data for the purpose of grouping. Subsequently, the grouped data works in the same way as local data binding virtualization, enhancing performance and responsiveness.

The following sample shows the example for Grouping with Virtualization.

INDEX.JS

```

var records = [];
for (var i = 1; i <= 150; i++) {
    var dropdownsItem = {};
    dropdownsItem.id = 'id' + i;
    dropdownsItem.text = "Item ".concat(i);
    var randomAutoGroup = Math.floor(Math.random() * 4) + 1;
    switch (randomAutoGroup) {
        case 1:
            dropdownsItem.group = 'Group D';
            break;
        case 2:
            dropdownsItem.group = 'Group C';
            break;
        case 3:
            dropdownsItem.group = 'Group B';
            break;
        case 4:
            dropdownsItem.group = 'Group A';
            break;
        default:
            break;
    }
    records.push(dropdownsItem);
}
//initiates the component
var acObject = new ej.dropdowns.AutoComplete({
    //bind the dataSorce property
    dataSource: records,
    //map the appropriate columns to fields property
    fields: { groupBy: 'group', value: 'text' },
    //set the placeholder to DropDownList input
    placeholder: "Select an Item ",
    //set enableVirtualization property to true
    enableVirtualization: true,
    //set the height of the popup element
    popupHeight: '200px'
});
//render the component

```

```
acObject.appendTo('#atcelement');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 AutoComplete</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:0 auto; width:250px;">
    <input type="text" id="atcelement">
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Localization in ##Platform_Name## Auto complete control

The Localization library allows you to localize static text content of the [noRecordsTemplate](#) and [actionFailureTemplate](#) properties according to the culture currently assigned to the AutoComplete.

| Locale key | en-US (default) |

|-----|-----|

| noRecordsTemplate | No Records Found |

| actionFailureTemplate | The Request Failed |

Loading translations

To load translation object to your application, use load function of the **L10n** class.

In the following sample, French culture is set to the AutoComplete and no data is loaded. Hence, the `noRecordsTemplate` property displays its text in French culture initially and if the sample is run offline, the `actionFailureTemplate` property displays its text appropriately.

INDEX.JS

```
//bind the data manager instance to dataSource property
var customerData = new ej.data.DataManager({
  url: 'https://services.odata.org/V4/Northwind/Northwind.svc/Customers',
  adaptor: new ej.data.ODataV4Adaptor(),
  crossDomain: true
});
//initiates the component
var acObj = new ej.dropdowns.AutoComplete({
  //set the data to dataSource property
  dataSource: customerData,
  // set locale culture to AutoComplete
  locale: 'fr-BE',
  // map appropriate column
  fields: { value: 'ContactName' },
  // take 0 item to showcase noRecordsTemplate property.
  query: new ej.data.Query().select(['ContactName',
  'CustomerID']).take(0),
  // set placeholder to AutoComplete input element
  placeholder: 'Sélectionnez un éléments'
});
acObj.appendTo('#atcelement');
ej.base.L10n.load({
  'fr-BE': {
    'dropdowns': {
      'noRecordsTemplate': "Aucun enregistrement trouvé",
      'actionFailureTemplate': "Modèle d'échec d'action"
    }
  }
});
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 AutoComplete</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin:0 auto; width:250px;">
        <br>
        <input type="text" id="atcelement">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Accessibility](#)
- [How to bind the data to the autocomplete](#)

Style in ##Platform_Name## Auto complete control

The following content provides the exact CSS structure that can be used to modify the control's appearance based on the user preference.

Customizing the appearance of wrapper element

Use the following CSS to customize the appearance of wrapper element.

```

.e-ddl.e-input-group.e-control-wrapper .e-input {
font-size: 20px;
font-family: emoji;
color: #ab3243;
background: #32a5ab;
}

```

Customizing the dropdown icon's color

Use the following CSS to customize the dropdown icon's color.

```

.e-ddl.e-input-group .e-input-group-icon,.e-ddl.e-input-group.e-control-wrapper .e-input-group-
icon:hover {
color: #bb233d;
font-size: 13px;
}

```

```
}
,
```

Customizing the focus color

Use the following CSS to customize the focusing color of input element.

```
,
```

```
.e-ddl.e-input-group.e-control-wrapper.e-input-focus::before, .e-ddl.e-input-group.e-control-wrapper.e-input-focus::after {
```

```
background: #c000ff;
```

```
}
```

```
,
```

Customizing the outline theme's focus color

Use the following CSS to customize the focusing color of outline theme.

```
,
```

```
.e-outline.e-input-group.e-input-focus:hover:not(.e-success):not(.e-warning):not(.e-error):not(.e-disabled):not(.e-float-icon-left),.e-outline.e-input-group.e-input-focus.e-control-wrapper:hover:not(.e-success):not(.e-warning):not(.e-error):not(.e-disabled):not(.e-float-icon-left),.e-outline.e-input-group.e-input-focus:not(.e-success):not(.e-warning):not(.e-error):not(.e-disabled),.e-outline.e-input-group.e-control-wrapper.e-input-focus:not(.e-success):not(.e-warning):not(.e-error):not(.e-disabled) {
```

```
border-color: #b1bd15;
```

```
box-shadow: inset 1px 1px #b1bd15, inset -1px 0 #b1bd15, inset 0 -1px #b1bd15;
```

```
}
```

```
,
```

Customizing the disabled component's text color

Use the following CSS to customize the text color when the component is disabled.

```
,
```

```
.e-input-group.e-control-wrapper .e-input[disabled] {
```

```
-webkit-text-fill-color: #0d9133;
```

```
}
```

```
,
```

Customizing the float label element's focusing color

Use the following CSS to customize the focusing color of float label element.

```
,
```

```
.e-float-input.e-input-group:not(.e-float-icon-left) .e-float-line::before,.e-float-input.e-control-wrapper.e-input-group:not(.e-float-icon-left) .e-float-line::before,.e-float-input.e-input-group:not(.e-float-icon-left) .e-float-line::after,.e-float-input.e-control-wrapper.e-input-group:not(.e-float-icon-left) .e-float-line::after {
```

```
background-color: #2319b8;
}
.e-ddl.e-input-group.e-control-wrapper.e-float-input.e-input-focus .e-float-text.e-label-top, .e-float-
input.e-control-wrapper:not(.e-error).e-input-focus input ~ label.e-float-text {
color: #2319b8;
}
、
```

Customizing the color of the placeholder text

Use the following CSS to customize the text color of placeholder.

```
、
.e-ddl.e-input-group input.e-input::placeholder {
color: red;
}
、
```

Customizing the text selection color

Use the following CSS to customize the selection color of text and background.

```
、
.e-ddl.e-input-group input.e-input::selection {
color: red;
background: yellow;
}
、
```

Customizing the background color of focus, hover, and active item's

Use the following CSS to customize the background color of focus, hover and active item's.

```
、
.e-dropdownbase .e-list-item.e-item-focus, .e-dropdownbase .e-list-item.e-active, .e-dropdownbase .e-
list-item.e-active.e-hover, .e-dropdownbase .e-list-item.e-hover {
background-color: #1f9c99;
color: #2319b8;
}
、
```

Customizing the appearance of pop-up element

Use the following CSS to customize the appearance of popup element.

```
、
.e-dropdownbase .e-list-item, .e-dropdownbase .e-list-item.e-item-focus {
```

```
background-color: #29c2b8;
color: #207cd9;
font-family: emoji;
min-height: 29px;
}
,
```

Adding mandatory asterisk to placeholder and float label

You can add a mandatory asterisk(*) to placeholder and float label using `.e-input-group.e-control-wrapper.e-float-input .e-float-text::after` class.

INDEX.JS

```
var sportsData = ['Badminton', 'Cricket', 'Football', 'Golf', 'Tennis'];
// initialize AutoComplete component
var listObj = new ej.dropdowns.AutoComplete({
  dataSource: sportsData,
  // set placeholder to AutoComplete input element
  placeholder: "Select games"});
listObj.appendTo('#atcelement');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 AutoComplete</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="asterisk.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:0 auto; width:250px;">
    <input type="text" tabindex="1" id="atcelement">
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
```

```

}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Accessibility in ##Platform_Name## Auto complete control

The AutoComplete component has been designed, keeping in mind the **WAI-ARIA** specifications, and applies the **WAI-ARIA** roles, states, and properties along with **keyboard support**. This component is characterized by complete keyboard interaction support and ARIA accessibility support that makes it easy for people who use assistive technologies (AT) or those who completely rely on keyboard navigation.

The AutoComplete component followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the AutoComplete component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2 Support](#) | |

| [Section 508 Support](#) | |

| [Screen Reader Support](#) | |

| [Right-To-Left Support](#) | |

| [Color Contrast](#) | |

| [Mobile Device Support](#) | |

| [Keyboard Navigation Support](#) | |

| [Accessibility Checker Validation](#) | |

| [Axe-core Accessibility Validation](#) | |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

```
</style>
```

```
<div> - All features of the component meet the requirement.</div>
```

```
<div> - Some features of the component do not meet the requirement.</div>
```

```
<div> - The component does not meet the requirement.</div>
```

WAI-ARIA attributes

The AutoComplete component uses the **combobox** role and each list item has an **option** role. The following ARIA Attributes denote the AutoComplete state.

| Property | Functionalities |

| --- | --- |

| aria-haspopup | Indicates whether the AutoComplete input element has a suggestion list or not. |

| aria-expanded | Indicates whether the suggestion list has expanded or not. |

| aria-selected | Indicates the selected option from the list. |

| aria-readonly | Indicates the readonly state of the AutoComplete element. |

| aria-disabled | Indicates whether the AutoComplete component is in a disabled state or not. |

| aria-activedescendent | This attribute holds the ID of the active list item to focus its descendant child element. |

| aria-owns | This attribute contains the ID of the suggestion list to indicate popup as a child element. |

| aria-autocomplete | This attribute contains the 'both' to a list of options shows and the currently selected suggestion also shows inline. |

Keyboard interaction

You can use the following key shortcuts to access the AutoComplete without interruptions.

| Keyboard shortcuts | Actions |

| --- | --- |

| Arrow Down | In popup hidden state, opens the suggestion list. In popup open state, selects the first item when no item selected else selects the item next to the currently selected item. |

| Arrow Up | In popup hidden state, opens the suggestion list. In popup open state, selects the last item when no item selected else selects the item previous to the currently selected one. |

| Page Down | Scrolls down to the next page and selects the first item when popup list opens. |

| Page Up | Scrolls up to previous page and select the first item when popup list open. |

| Enter | Selects the focused item and set to AutoComplete component. |

| Tab | Focuses on the next tab indexed element when the popup is closed. Otherwise, closes the popup list and remains the focus in component suppose if it is in an open state. |

| Shift + tab | Focuses the previous tab indexed element when the popup is closed. Otherwise, closes the popup list and remains the focus in component suppose if it is in an open state. |

| Alt + Down | Opens the popup list. |

| Alt + Up | In popup hidden state, opens the popup list. In popup open state, closes the popup list. |

| Esc(Escape) | Closes the popup list when it is in an open state then remove the selection. |

| Home | Cursor moves to before of first character in input. |

| End | Cursor moves to next of last character in input. |

In the below sample, focus the AutoComplete component using alt+t keys.

INDEX.JS

```
var gameList = [
    { id: 'Game1', game: 'Badminton' },
    { id: 'Game2', game: 'Basketball' },
    { id: 'Game3', game: 'Cricket' },
    { id: 'Game4', game: 'Football' },
    { id: 'Game5', game: 'Golf' },
    { id: 'Game6', game: 'Hockey' },
    { id: 'Game7', game: 'Rugby' },
    { id: 'Game8', game: 'Snooker' },
    { id: 'Game9', game: 'Tennis' },
];
// initialize AutoComplete component
var acObject = new ej.dropdowns.AutoComplete({
    //set the data to dataSource property
    dataSource: gameList,
    //map to colum to fields
    fields: { value: 'game' },
    // set placeholder to AutoComplete input element
    placeholder: "Select games",
    // set the popup list height
    popupHeight: '200px'
});
// render initialized AutoComplete
acObject.appendTo('#atcelement');
document.onkeyup = function (e) {
    if (e.altKey && e.keyCode === 84 /* t */) {
        // press alt+t to focus the control.
        acObject.focusIn();
    }
};
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 AutoComplete</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
```



```

<meta name="author" content="Syncfusion">
<link href="styles.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin:0 auto; width:250px;">
        <input type="text" tabindex="1" id="atcelement">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Ensuring accessibility

The AutoComplete component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the AutoComplete component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the AutoComplete component with accessibility tools.

See also

- [Accessibility in Syncfusion components](#)

How To

Autofill in ##Platform_Name## Auto complete control

The AutoComplete supports the autofill behavior with the help of [autofill](#) property. Whenever you change the input value, the AutoComplete will autocomplete your data by matching the typed character. Suppose, if no matches found then, AutoComplete doesn't suggest any item.

In the below sample, showcase that how to work [autofill](#) with AutoComplete.

INDEX.JS

```

let searchData = [
    { Name: 'Australia', Code: 'AU' },
    { Name: 'Bermuda', Code: 'BM' },

```

```

    { Name: 'Canada', Code: 'CA' },
    { Name: 'Cameroon', Code: 'CM' },
    { Name: 'Denmark', Code: 'DK' },
    { Name: 'France', Code: 'FR' },
    { Name: 'Finland', Code: 'FI' },
    { Name: 'Germany', Code: 'DE' },
    { Name: 'Greenland', Code: 'GL' },
    { Name: 'Hong Kong', Code: 'HK' },
    { Name: 'India', Code: 'IN' },
    { Name: 'Italy', Code: 'IT' },
    { Name: 'Japan', Code: 'JP' },
    { Name: 'Mexico', Code: 'MX' },
    { Name: 'Norway', Code: 'NO' },
    { Name: 'Poland', Code: 'PL' },
    { Name: 'Switzerland', Code: 'CH' },
    { Name: 'United Kingdom', Code: 'GB' },
    { Name: 'United States', Code: 'US' }
  ];
  //initiates the component
  let atcObject = new ej.dropdowns.AutoComplete({
    // bind the country data to dataSource property
    dataSource: searchData,
    // maps the appropriate column to fields property
    fields: { value: "Name" },
    //set the placeholder to AutoComplete input
    placeholder: "Find a country",
    //enable the autofill property to fill a first matched value in input
    //when press a down key
    autofill: true
  });
  atcObject.appendTo("#atcelement");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 AutoComplete</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

```

```

<div id="container" style="margin:0 auto; width:250px;">
  <input type="text" tabindex="1" id="atcelement">
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Icon support in ##Platform_Name## Auto complete control

You can render **icons** to the list items by mapping the `iconCss` field. This `iconCss` field create a span in the list item with mapped class name to allow styling as per your need.

In the following sample, the icon classes are mapped with `iconCss` field.

INDEX.JS

```

let sortFormatData = [
  { Class: 'asc-sort', Type: 'Sort A to Z', Id: '1' },
  { Class: 'dsc-sort', Type: 'Sort Z to A ', Id: '2' },
  { Class: 'filter', Type: 'Filter', Id: '3' },
  { Class: 'clear', Type: 'Clear', Id: '4' }
];
//initiates the component
let sortFormat = new ej.dropdowns.AutoComplete({
  //set the data to dataSource property
  dataSource: sortFormatData,
  // map the icon column to iconCSS field.
  fields: { value: 'Type', iconCss: 'Class' },
  // set placeholder to AutoComplete input element
  placeholder: 'Find a format'
});
sortFormat.appendTo('#atcelement');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 AutoComplete</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin:0 auto; width:250px;">
        <br>
        <input type="text" id="atcelement">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Custom search in ##Platform_Name## Auto complete control

The AutoComplete has built-in support to highlight the searched characters on suggested list items when enabled the [highlight](#) property.

In the below sample, to customize the matched character in suggestion list by `e-highlight` class.

INDEX.JS

```

let searchData = [
    { Name: 'Australia', Code: 'AU' },
    { Name: 'Bermuda', Code: 'BM' },
    { Name: 'Canada', Code: 'CA' },
    { Name: 'Cameroon', Code: 'CM' },
    { Name: 'Denmark', Code: 'DK' },
    { Name: 'France', Code: 'FR' },
    { Name: 'Finland', Code: 'FI' },
    { Name: 'Germany', Code: 'DE' },
    { Name: 'Greenland', Code: 'GL' },
    { Name: 'Hong Kong', Code: 'HK' },
    { Name: 'India', Code: 'IN' },
    { Name: 'Italy', Code: 'IT' },
    { Name: 'Japan', Code: 'JP' },
    { Name: 'Mexico', Code: 'MX' },
    { Name: 'Norway', Code: 'NO' },
    { Name: 'Poland', Code: 'PL' },
    { Name: 'Switzerland', Code: 'CH' },
    { Name: 'United Kingdom', Code: 'GB' },
    { Name: 'United States', Code: 'US' }
];
//initiates the component
let atcObject = new ej.dropdowns.AutoComplete({
    // bind the country data to dataSource property
    dataSource: searchData,
    // maps the appropriate column to fields property
    fields: { value: "Name" },
    //set the placeholder to AutoComplete input

```

```

    placeholder: "Find a country",
    //enable the highlight property to highlight the matched character in
    suggestion list
    highlight: true
  });
  atcObject.appendTo('#atcelement');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 AutoComplete</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:0 auto; width:250px;">
    <input type="text" tabindex="1" id="atcelement">
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Filter in ##Platform_Name## Auto complete control

The AutoComplete data can be filtered based on both text and value fields using predicate of dataManager through filtering event. The filtered data can be again updated through updateData method.

In the following example, filtering is done based on text and value fields.

INDEX.JS

```

let searchData = [
  { Name: 'Australia', Code: 'AU' },
  { Name: 'Bermuda', Code: 'BM' },

```

```

    { Name: 'Canada', Code: 'CA' },
    { Name: 'Cameroon', Code: 'CM' },
    { Name: 'Denmark', Code: 'DK' },
    { Name: 'France', Code: 'FR' },
    { Name: 'Finland', Code: 'FI' },
    { Name: 'Germany', Code: 'DE' },
    { Name: 'Greenland', Code: 'GL' },
    { Name: 'Hong Kong', Code: 'HK' },
    { Name: 'India', Code: 'IN' },
    { Name: 'Italy', Code: 'IT' },
    { Name: 'Japan', Code: 'JP' },
    { Name: 'Mexico', Code: 'MX' },
    { Name: 'Norway', Code: 'NO' },
    { Name: 'Poland', Code: 'PL' },
    { Name: 'Switzerland', Code: 'CH' },
    { Name: 'United Kingdom', Code: 'GB' },
    { Name: 'United States', Code: 'US' }
  ];
  //initiates the component
  let atcObject = new ej.dropdowns.AutoComplete({
    // bind the country data to dataSource property
    dataSource: searchData,
    // maps the appropriate column to fields property
    fields: { value: "Code", text:"Name" },
    //set the placeholder to AutoComplete input
    //set item template
    itemTemplate:"<span><span class='name'>${Name}</span><span class
='code'>${Code}</span></span>",
    filtering: function(e)
    {
      e.preventDefaultAction=true;
      var predicate = new ej.data.Predicate('Name', 'contains', e.text);
      predicate = predicate.or('Code', 'contains', e.text);
      var query = new ej.data.Query();
      //frame the query based on search string with filter type.
      query = (e.text != "") ? query.where(predicate) : query;
      //pass the filter data source, filter query to updateData method.
      e.updateData(this.dataSource, query);
    }
  });
  atcObject.appendTo('#atcelement');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 AutoComplete</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin:0 auto; width:250px;">
        <input type="text" tabindex="1" id="atcelement">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Achieve virtual scrolling in ##Platform_Name## Auto complete control

The Virtual Scrolling is used to display a large amount of data without buffering the entire load of a huge database record in the AutoComplete, that is, when scrolling, the request is sent and fetch some amount of data from the server dynamically. Using the `scroll` event, get the data and generate the list add to popup using the `addItem` method.

Refer to the following code sample for virtual scrolling.

INDEX.JS

```

var data = new ej.data.DataManager({
    url: 'https://js.syncfusion.com/demos/ejServices/Wcf/Northwind.svc/',
    crossDomain: true
});
//initiates the component
let atcObject = new ej.dropdowns.AutoComplete({
    // bind the DataManager instance to dataSource property
    dataSource: data,
    // bind the Query instance to query property
    query: new
ej.data.Query().from('Customers').select('ContactName').take(7),
    // map the appropriate columns to fields property
    fields: { text: 'ContactName', value: 'ContactName' },
    // set the placeholder to DropDownList input element
    placeholder: 'Select a customer',
    // sort the resulted items
    sortOrder: 'Ascending',
    // set the height of the popup element
    popupHeight: '200px',

    actionComplete: function (e) {
        let operator = new
ej.data.Query().from('Customers').select('ContactName');

```

```

        let star = 7;
        let end = 12;
        let listElement = this.list;
        listElement.addEventListener('scroll', () => {
            if ((listElement.scrollTop + listElement.offsetHeight >=
listElement.scrollHeight)) {
                let filterQuery = operator.clone();
                data.executeQuery(filterQuery.range(start,
end)).then((event) => {
                    start = end;
                    end += 5;
                    atcObject.addItem(event.result);
                }).catch((e) => {
                });
            }
        })
    });
    atcObject.appendTo('#atcelement');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 AutoComplete</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin:0 auto; width:250px;">
        <input type="text" tabindex="1" id="atcelement">
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```


Avatar

Types in ##Platform_Name## Avatar control

This section explains different types of avatar.

Avatar size

The Essential JS 2 Avatar has the following predefined sizes that can be used with the `.e-avatar` class to change the appearance of the avatar.

Class Name	Description
:-----	:-----
e-avatar-xlarge	Displays xlarge size avatar.
e-avatar-large	Displays apply large size avatar.
e-avatar	Displays apply default size avatar.
e-avatar-small	Displays apply small size avatar.
e-avatar-xsmall	Displays apply xsmall size avatar.

INDEX.JS

--

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for Avatar </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for Avatar UI Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element">
      <span class="e-avatar e-avatar-xlarge"></span>
      <span class="e-avatar e-avatar-large"></span>
      <span class="e-avatar"></span>
      <span class="e-avatar e-avatar-small"></span>
      <span class="e-avatar e-avatar-xsmall"></span>
    </div>
  </div>
```

```

    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Avatar types

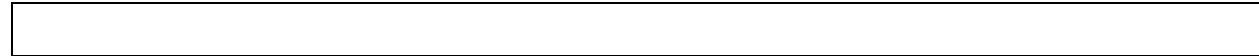
The types of Essential JS 2 avatar are:

- Default
- Circle

Default

The default style of the avatar is rectangular shape with rounded corners, which can be applied from adding the modifier class `.e-avatar` to the target element.

INDEX.JS



INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for Avatar </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for Avatar UI Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element">
      <span class="e-avatar e-avatar-xlarge">RT</span>
      <span class="e-avatar e-avatar-large">RT</span>
      <span class="e-avatar">RT</span>
      <span class="e-avatar e-avatar-small">RT</span>
      <span class="e-avatar e-avatar-xsmall">RT</span>
    </div>
  </div>

```

```

        </div>
    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Circle

The circle avatar style can be applied by adding the modifier class `.e-avatar-circle` with default avatar modifier class `.e-avatar` to the target element.

INDEX.JS

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 for Avatar </title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Essential JS 2 for Avatar UI Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element">
            <span class="e-avatar e-avatar-xlarge e-avatar-circle">SJ</span>
            <span class="e-avatar e-avatar-large e-avatar-circle">SJ</span>
            <span class="e-avatar e-avatar-circle">SJ</span>
            <span class="e-avatar e-avatar-small e-avatar-circle">SJ</span>
            <span class="e-avatar e-avatar-xsmall e-avatar-circle">SJ</span>
        </div>
    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}

```

```

    </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

How To

Avatar customization in ##Platform_Name## Avatar control

Color customization

The avatar comes with default background color (grey). This can be easily customized to desired color by adding custom class or directly selecting the avatar class from the CSS.

INDEX.JS

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for Avatar </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for Avatar UI Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element">
      <span class="e-avatar e-avatar-xlarge e-avatar-circle
green">AJ</span>
      <span class="e-avatar e-avatar-xlarge e-avatar-circle
violet">JK</span>
      <span class="e-avatar e-avatar-xlarge e-avatar-circle
rose">EL</span>
      <span class="e-avatar e-avatar-xlarge e-avatar-circle
blue">SR</span>
      <span class="e-avatar e-avatar-xlarge e-avatar-circle
red">PD</span>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";

```

```

}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customize avatar sizes

Even though the avatar comes with five predefined sizes, sometimes it's not enough. So, the avatar is designed in such a way that the width and height will be relative to font-size. By changing the font-size of the avatar element, you can change the width and height automatically.

INDEX.JS

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for Avatar </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for Avatar UI Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">

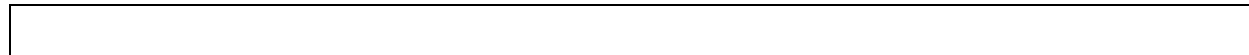
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element">
      <span class="e-avatar">26px</span>
      <span class="e-avatar">24px</span>
      <span class="e-avatar">22px</span>
      <span class="e-avatar">20px</span>
      <span class="e-avatar">18px</span>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Use various media in avatar

Avatars can be used with a wide variety of media formats like SVG, font-icons, images, letters, words, etc. Some of them are given below.

INDEX.JS**INDEX.HTML**

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for Avatar </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for Avatar UI Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element">
      <div class="sample_container avatar-types">
        <div class="avatar-block">
          <!-- Card Component -->
          <div class="e-card e-avatar-showcase">
            <div class="e-card-content">
              <!-- XLarge Circle Avatar Component -->
              <div class="e-avatar e-avatar-xlarge e-avatar-
circle image"></div>

              </div>
              <div class="e-card-content">
                <div>Image</div>
              </div>
            </div>
          </div>
          <div class="avatar-block">
            <!-- Card Component -->
            <div class="e-card e-avatar-showcase">
              <div class="e-card-content">
                <!-- XLarge Circle Avatar Component -->
                <div class="e-avatar e-avatar-xlarge e-avatar-
circle">

                  <div class="svg_icons chrome"></div>
                </div>
              </div>
            </div>
          </div>
        </div>
      </div>
    </div>
  </div>

```

```

        <div class="e-card-content">
            <div>SVG</div>
        </div>
    </div>
</div>
<div class="avatar-block">
    <!-- Card Component -->
    <div class="e-card e-avatar-showcase">
        <div class="e-card-content">
            <!-- XLarge Circle Avatar Component -->
            <div class="e-avatar e-avatar-xlarge e-avatar-
circle">GR</div>
        </div>
        <div class="e-card-content">
            <div>Initial</div>
        </div>
    </div>
</div>
<div class="avatar-block">
    <!-- Card Component -->
    <div class="e-card e-avatar-showcase">
        <div class="e-card-content">
            <!-- XLarge Circle Avatar Component -->
            <div class="e-avatar e-avatar-xlarge e-avatar-
circle">
                <div class="e-people icons"></div>
            </div>
        </div>
        <div class="e-card-content">
            <div>FontIcon</div>
        </div>
    </div>
</div>
<div class="avatar-block">
    <!-- Card Component -->
    <div class="e-card e-avatar-showcase">
        <div class="e-card-content">
            <!-- XLarge Circle Avatar Component -->
            <div class="e-avatar e-avatar-xlarge e-avatar-
circle">User</div>
        </div>
        <div class="e-card-content">
            <div>Word</div>
        </div>
    </div>
</div>
<div class="avatar-block">
    <!-- Card Component -->
    <div class="e-card e-avatar-showcase">
        <div class="e-card-content">
            <!-- XLarge Circle Avatar Component -->
            <div class="e-avatar e-avatar-xlarge e-avatar-
circle custom">
                <div class="e-people icons"></div>
            </div>
        </div>
        <div class="e-card-content">

```

```

        <div>Custom</div>
    </div>
</div>
</div>
</div>
</div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Integrate avatar into listview in ##Platform Name## Avatar control

Avatar can be integrated into various components to make a wide variety of applications. Some of the integrations are shown in the following section.

Avatar is integrated into the listview to create contacts applications. The `xsmall` size avatar is used to match the size of the list item. Letters and images are also used as avatar content.

INDEX.JS

```
var dataSource = [
  { id: 's_01', text: 'Robert', avatar: '', pic: 'pic04' },
  { id: 's_02', text: 'Nancy', avatar: 'N', pic: '' },
  { id: 's_05', text: 'John', pic: 'pic01', avatar: '' },
  { id: 's_03', text: 'Andrew', avatar: 'A', pic: '' },
  { id: 's_06', text: 'Michael', pic: 'pic02', avatar: '' },
  { id: 's_07', text: 'Steven', pic: 'pic03', avatar: '' },
  { id: 's_08', text: 'Margaret', avatar: 'M', pic: '' }
];

var letterAvatarList = new ej.lists.ListView({
  // Bind listview datasource
  dataSource: dataSource,
  // Assign header title
  headerTitle: 'Contacts',
  // Enable header title
  showHeader: true,
  // Assign list-item template
  template: '<div class="listWrapper">' +
    '${if(avatar!=="")}' +
    '<span class="e-avatar e-avatar-small e-avatar-circle">${avatar}</span>' +
    '${else}' +
    '<span class="${pic} e-avatar e-avatar-small e-avatar-circle">' +
    '</span>' +
    '${/if}' +
    '<span class="text">' +
    '${text}' +
    '</span> </div>',
  // Assign sorting order
  sortOrder: 'Ascending'
});

letterAvatarList.appendTo('#letterAvatarList');
```


INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for Avatar </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for Avatar UI Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element">
      <!-- Listview element -->
      <div id="letterAvatarList"></div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Integrate avatar into badge in ##Platform_Name## Avatar control

The badge is dependent and supportive component, and it can be used with avatar to create a notification avatar.

The default avatar (.e-avatar) and circle avatar (.e-avatar-circle) have been used with notification badges (.e-badge-notification) in the following sample.

INDEX.JS

```


```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>

```

```

<title>Essential JS 2 for Avatar </title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Essential JS 2 for Avatar UI Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element">
            <div class="sample_container avatar-badge">
                <div class="avatar-block">
                    <!-- Card Component -->
                    <div class="e-card e-avatar-showcase">
                        <div class="e-card-content">
                            <div class="avatar-sub-block">
                                <!-- xSmall Avatar-->
                                <div class="e-avatar e-avatar-xsmall">
                                    

                                </div>
                                <!-- Notification Badge -->
                                <span class="e-badge e-badge-primary e-
badge-overlap e-badge-notification e-badge-circle">6</span>
                                </div>
                                <div class="avatar-sub-block">
                                    <!-- Small Avatar-->
                                    <div class="e-avatar e-avatar-small">
                                        

                                    </div>
                                    <!-- Notification Badge -->
                                    <span class="e-badge e-badge-primary e-
badge-overlap e-badge-notification e-badge-circle">12</span>
                                    </div>
                                    <div class="avatar-sub-block">
                                        <!-- Avatar-->
                                        <div class="e-avatar">
                                            

                                        </div>
                                        <!-- Notification Badge -->
                                        <span class="e-badge e-badge-primary e-
badge-overlap e-badge-notification">46</span>

```

```

        </div>
        <div class="avatar-sub-block">
            <!-- Large Avatar-->
            <div class="e-avatar e-avatar-large">
                
            </div>
            <!-- Notification Badge -->
            <span class="e-badge e-badge-primary e-
badge-overlap e-badge-notification">82</span>
            </div>
            <div class="avatar-sub-block">
                <!-- xLarge Avatar-->
                <div class="e-avatar e-avatar-xlarge">
                    
                </div>
                <!-- Notification Badge -->
                <span class="e-badge e-badge-primary e-
badge-overlap e-badge-notification">99+</span>
            </div>
        </div>
    </div>
</div>
<div class="circleAvatar avatar-block">
    <!-- Card Component -->
    <div class="e-card e-avatar-showcase">
        <div class="e-card-content">
            <div class="avatar-sub-block">
                <!-- xSmall Circle Avatar-->
                <div class="e-avatar e-avatar-circle e-
avatar-xsmall">
                    
                </div>
                <!-- Notification Badge -->
                <span class="e-badge e-badge-primary e-
badge-overlap e-badge-notification e-badge-circle">6</span>
            </div>
            <div class="avatar-sub-block">
                <!-- Small Circle Avatar-->
                <div class="e-avatar e-avatar-circle e-
avatar-small">
                    
                </div>
                <!-- Notification Badge -->
                <span class="e-badge e-badge-primary e-
badge-overlap e-badge-notification e-badge-circle">12</span>
            </div>
            <div class="avatar-sub-block">
                <!-- Circle Avatar-->
                <div class="e-avatar e-avatar-circle">
                    
                </div>
                <!-- Notification Badge -->
            </div>
        </div>
    </div>
</div>

```

```
<span class="e-badge e-badge-primary e-  
badge-overlap e-badge-notification">46</span>  
</div>  
<div class="avatar-sub-block">  
<!-- Large Circle Avatar-->  
<div class="e-avatar e-avatar-circle e-  
avatar-large">  
  
      
  
</div>  
<!-- Notification Badge -->  
<span class="e-badge e-badge-primary e-  
badge-overlap e-badge-notification">82</span>  
</div>  
<div class="avatar-sub-block">  
<!-- xLarge Circle Avatar-->  
<div class="e-avatar e-avatar-circle e-  
avatar-xlarge">  
  
      
  
</div>  
<!-- Notification Badge -->  
<span class="e-badge e-badge-primary e-  
badge-overlap e-badge-notification">99+</span>  
</div>  
</div>  
</div>  
</div>  
</div>  
</div>  
</div>  
<script>  
var ele = document.getElementById('container');  
if(ele) {  
    ele.style.visibility = "visible";  
}  
  
</script>  
<script src="index.js" type="text/javascript"></script>  
</body></html>
```

Badge

Types in ##Platform Name## Badge control

This section explains different styles and types of the badges.

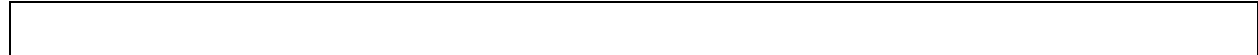
Badge styles

The Essential JS 2 Badge has the following predefined styles that can be used with `.e-badge` class to change the appearance of a badge.

Class Name	Description
e-badge-primary	Represents a primary notification.
e-badge-secondary	Represents a secondary notification.

- | e-badge-success | Represents a positive notification.
- | e-badge-danger | Represents a negative notification.
- | e-badge-warning | Represents notification with caution.
- | e-badge-info | Represents an informative notification.
- | e-badge-light | Represents notification in light variant.
- | e-badge-dark | Represents notification in dark variant.

INDEX.JS



INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for Badge </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for Badge UI Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element">
      <div class="sample_container">
        <div class="block" style="display:inline-block">
          <div class="e-card e-badge-showcase">
            <div class="e-card-content">
              <div>
                <span class="e-badge e-badge-
primary">Primary</span>
              </div>
            </div>
            <div class="e-card-content">
              <div>
                <code>.e-badge-primary</code>
              </div>
            </div>
          </div>
        </div>
      </div>
    </div>
  </div>
```

```

        </div>
        <div class="block" style="display:inline-block">
            <div class="e-card e-badge-showcase">
                <div class="e-card-content">
                    <span class="e-badge e-badge-
secondary">Secondary</span>
                </div>
                <div class="e-card-content">
                    <code>.e-badge-secondary</code>
                </div>
            </div>
        </div>
        <div class="block" style="display:inline-block">
            <div class="e-card e-badge-showcase">
                <div class="e-card-content">
                    <span class="e-badge e-badge-
success">Success</span>
                </div>
                <div class="e-card-content">
                    <code>.e-badge-success</code>
                </div>
            </div>
        </div>
        <div class="block" style="display:inline-block">
            <div class="e-card e-badge-showcase">
                <div class="e-card-content">
                    <span class="e-badge e-badge-
danger">Danger</span>
                </div>
                <div class="e-card-content">
                    <code>.e-badge-danger</code>
                </div>
            </div>
        </div>
        <div class="block" style="display:inline-block">
            <div class="e-card e-badge-showcase">
                <div class="e-card-content">
                    <span class="e-badge e-badge-
warning">Warning</span>
                </div>
                <div class="e-card-content">
                    <code>.e-badge-warning</code>
                </div>
            </div>
        </div>
        <div class="block" style="display:inline-block">
            <div class="e-card e-badge-showcase">
                <div class="e-card-content">
                    <span class="e-badge e-badge-info">Info</span>
                </div>
                <div class="e-card-content">
                    <code>.e-badge-info</code>
                </div>
            </div>
        </div>
        <div class="block" style="display:inline-block">
            <div class="e-card e-badge-showcase">

```

```

        <div class="e-card-content">
            <span class="e-badge e-badge-light">Light</span>
        </div>
        <div class="e-card-content">
            <code>.e-badge-light</code>
        </div>
    </div>
</div>
<div class="block" style="display:inline-block">
    <div class="e-card e-badge-showcase">
        <div class="e-card-content">
            <span class="e-badge e-badge-dark">Dark</span>
        </div>
        <div class="e-card-content">
            <code>.e-badge-dark</code>
        </div>
    </div>
</div>
</div>
</div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Badge types

The types of Essential JS 2 badges are as follows:

- Circle
- Pill
- Link
- Notification
- Overlap
- Dot
- Position

Circle

The circle badge style can be applied by adding the modifier class `.e-badge-circle` to the target element.

INDEX.JS

--

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for Badge </title>
  <meta charset="utf-8">
```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Essential JS 2 for Badge UI Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element">
            <div style="position:relative;display:inline-block;margin:20px
20px 10px 20px;">
                <div class="skype svg_icons"></div>
                <span class="e-badge e-badge-success e-badge-overlap e-
badge-notification e-badge-circle">18</span>
            </div>
            <div style="position:relative;display:inline-block;margin:20px
20px 10px 20px;">
                <div class="twitter svg_icons"></div>
                <span class="e-badge e-badge-info e-badge-overlap e-badge-
notification e-badge-circle">9</span>
            </div>
            <div style="position:relative;display:inline-block;margin:20px
20px 10px 20px;">
                <div class="facebook svg_icons"></div>
                <span class="e-badge e-badge-info e-badge-overlap e-badge-
notification e-badge-circle">2</span>
            </div>
            <div style="position:relative;display:inline-block;margin:20px
20px 10px 20px;">
                <div class="firefox svg_icons"></div>
                <span class="e-badge e-badge-danger e-badge-overlap e-badge-
notification e-badge-circle">35</span>
            </div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```


Pill

The pill badge style can be applied by adding the modifier class `e-badge-pill` to the target element.

INDEX.JS



INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for Badge </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for Badge UI Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">

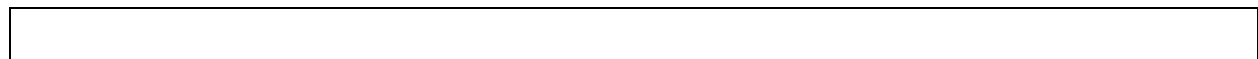
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element">
      <h1>Badge Component <span class="e-badge e-badge-primary e-
badge-pill">New</span></h1>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Link

When badge modifier classes are applied to the anchor tag, the badge's appearance will change from normal state to hover state on mouseover.

INDEX.JS



INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for Badge </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for Badge UI Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element">
      <div style="display: inline-block; margin-top: 15px;">
        <a href="#" class="e-badge e-badge-primary">Link Badge</a>
      </div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Notification

The notification badge style can be applied by adding the modifier class `.e-badge-notification` to the target element. Notification badges are used when a content or a context needs special attention. While using the notification badge, set the parent element to `position: relative`.

INDEX.JS

```

// Empty content for INDEX.JS

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for Badge </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for Badge UI Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element">
            <div style="position:relative;display:inline-block;margin:20px
20px 10px 20px;">
                <div class="skype svg_icons"></div>
                <span class="e-badge e-badge-success e-badge-overlap e-
badge-notification">99+</span>
            </div>
            <div style="position:relative;display:inline-block;margin:20px
20px 10px 20px;">
                <div class="twitter svg_icons"></div>
                <span class="e-badge e-badge-info e-badge-overlap e-badge-
notification">27</span>
            </div>
            <div style="position:relative;display:inline-block;margin:20px
20px 10px 20px;">
                <div class="facebook svg_icons"></div>
                <span class="e-badge e-badge-info e-badge-overlap e-badge-
notification">2</span>
            </div>
            <div style="position:relative;display:inline-block;margin:20px
20px 10px 20px;">
                <div class="firefox svg_icons"></div>
                <span class="e-badge e-badge-danger e-badge-overlap e-badge-
notification">35</span>
            </div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Dot

Dot can be applied by adding the modifier class `.e-badge-dot` to the target element. Dot badges are similar to notification badges, but in a minimalistic way. While using the dot badge, set the parent element to `position: relative`.

INDEX.JS

--

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for Badge </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for Badge UI Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element">
      <div style="position:relative;display:inline-block;margin:20px
20px 10px 20px;">
        <div class="skype svg_icons"></div>
        <span class="e-badge e-badge-success e-badge-overlap e-
badge-dot"></span>
      </div>
      <div style="position:relative;display:inline-block;margin:20px
20px 10px 20px;">
        <div class="twitter svg_icons"></div>
        <span class="e-badge e-badge-info e-badge-overlap e-badge-
dot"></span>
      </div>
      <div style="position:relative;display:inline-block;margin:20px
20px 10px 20px;">
        <div class="facebook svg_icons"></div>
        <span class="e-badge e-badge-info e-badge-overlap e-badge-
dot"></span>
      </div>
      <div style="position:relative;display:inline-block;margin:20px
20px 10px 20px;">
        <div class="firefox svg_icons"></div>
        <span class="e-badge e-badge-danger e-badge-overlap e-badge-
dot"></span>
      </div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');

```

```

if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Overlap

The overlap badge can be used with notification or dot badge, which overlaps with the target element by adding the modifier class `e-badge-overlap`. While using the overlap badge, set the parent element to `position: relative`.

INDEX.JS

```


```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for Badge </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for Badge UI Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element">
      <div style="position:relative;display:inline-block;margin:20px
20px 10px 20px;">
        <div class="skype svg_icons"></div>
        <span class="e-badge e-badge-success e-badge-overlap e-
badge-notification">99+</span>
      </div>
      <div style="position:relative;display:inline-block;margin:20px
20px 10px 20px;">
        <div class="twitter svg_icons"></div>
        <span class="e-badge e-badge-info e-badge-overlap e-badge-
notification">27</span>
      </div>
      <div style="position:relative;display:inline-block;margin:20px
20px 10px 20px;">

```

```

        <div class="facebook svg_icons"></div>
        <span class="e-badge e-badge-info e-badge-overlap e-badge-
notification">2</span>
    </div>
    <div style="position:relative;display:inline-block;margin:20px
20px 10px 20px;">
        <div class="firefox svg_icons"></div>
        <span class="e-badge e-badge-danger e-badge-overlap e-badge-
notification">35</span>
    </div>
</div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Position

The default position of the notification or dot badge is top. But, the position can be changed to **bottom** using the modifier class **.e-badge-bottom**. For example, the bottom class modifier is used with dot badge to display the status in the avatar as shown in the following sample.

INDEX.JS

```


```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 for Badge </title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Essential JS 2 for Badge UI Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element">

```

```

        <div class="badge-block">
            <div class="contact svg_icons"></div>
            <!-- Success Colored Bottom Dot Badge -->
            <span class="e-badge e-badge-success e-badge-overlap e-
badge-dot e-badge-bottom"></span>
        </div>
        <div class="badge-block">
            <div class="skype svg_icons"></div>
            <!-- Info Colored Bottom Dot Badge -->
            <span class="e-badge e-badge-info e-badge-overlap e-badge-
dot e-badge-bottom"></span>
        </div>
        <div class="badge-block">
            <div class="facebook svg_icons"></div>
            <!-- Info Colored Dot Badge -->
            <span class="e-badge e-badge-info e-badge-overlap e-badge-
dot"></span>
        </div>
        <div class="badge-block">
            <div class="pinterest svg_icons"></div>
            <!-- Danger Colored Dot Badge -->
            <span class="e-badge e-badge-danger e-badge-overlap e-badge-
dot e-badge-bottom"></span>
        </div>
    </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

How To

Badge customization in **##Platform_Name##** Badge control

Colour customization

Even though badges come with **8 predefined colors**, you can also customize the colour of the badge as desired.

INDEX.JS

```


```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 for Badge </title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Essential JS 2 for Badge UI Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element">
            <h1>Color Customization <span class="e-badge e-badge-primary e-
badge-pill green">New</span></h1>
            <h1>Color Customization <span class="e-badge e-badge-primary e-
badge-pill blue">New</span></h1>
            <h1>Color Customization <span class="e-badge e-badge-primary e-
badge-pill purple">New</span></h1>
            <h1>Color Customization <span class="e-badge e-badge-primary e-
badge-pill gradient">New</span></h1>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customize badge size

Badges are designed to change its size based on the content. To change the size of a badge, adjust the font size of the badge.

INDEX.JS

```


```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 for Badge </title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Essential JS 2 for Badge UI Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">

```



```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element">
            <h1>Badge Component <span class="e-badge e-badge-primary
size_1">New</span></h1>
            <h1>Badge Component <span class="e-badge e-badge-primary
size_2">New</span></h1>
            <h1>Badge Component <span class="e-badge e-badge-primary
size_3">New</span></h1>
        </div>
    </div>
</body>
</html>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>

```

Custom position

Even though the badges support the conventional **top** and **bottom** positions, the position of the badges can be changed as desired.

This can be done by adding a custom class to the badge element to override the default position applied from the source.

INDEX.JS

```

// Empty content for INDEX.JS

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 for Badge </title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Essential JS 2 for Badge UI Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element">
            <div style="width: 200px;margin: 10px auto;">
                <div class="badge-block">
                    <div class="whatsapp svg_icons"></div>
                    <!-- Warning Colored Notification Badge -->
                    <span class="e-badge e-badge-warning e-badge-
notification e-badge-overlap leftTop">99+</span>
                </div>
                <div class="badge-block">
                    <div class="facebook svg_icons"></div>
                    <!-- Danger Colored Notification Badge -->
                    <span class="e-badge e-badge-danger e-badge-notification
e-badge-overlap leftTop">99+</span>
                </div>
                <div class="badge-block">
                    <div class="skype svg_icons"></div>
                    <!-- Secondary Colored Notification Badge -->
                    <span class="e-badge e-badge-secondary e-badge-
notification e-badge-overlap leftTop">18</span>
                </div>
            </div>
            <div style="width: 200px;margin: 10px auto;">
                <div class="badge-block">
                    <div class="whatsapp svg_icons"></div>
                    <!-- Warning Colored Notification Badge -->
                    <span class="e-badge e-badge-warning e-badge-
notification e-badge-overlap leftBottom">99+</span>
                </div>
                <div class="badge-block">
                    <div class="facebook svg_icons"></div>
                    <!-- Danger Colored Notification Badge -->
                    <span class="e-badge e-badge-danger e-badge-notification
e-badge-overlap leftBottom">99+</span>
                </div>
                <div class="badge-block">
                    <div class="skype svg_icons"></div>
                    <!-- Secondary Colored Notification Badge -->
                    <span class="e-badge e-badge-secondary e-badge-
notification e-badge-overlap leftBottom">18</span>
                </div>
            </div>
        </div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>

```

```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Integrate badge into listview in ##Platform_Name## Badge control

The badges can be integrated with the `listview` component to indicate new notification with colour based on priority.

In the following sample, `default` badges are used and there is no need to customize the badge size. The component will automatically adjust the size based on the container element.

INDEX.JS

```
// Datasource for listview, badge field is the class data for Badges
var dataSource = [
  { id: 'p_01', text: 'Primary', messages: '3 New', badge: 'e-badge e-
badge-primary', icons: 'primary', type: 'Primary' },
  { id: 'p_02', text: 'Social', messages: '27 New', badge: 'e-badge e-
badge-secondary', icons: 'social', type: 'Primary' },
  { id: 'p_03', text: 'Promotions', messages: '7 New', badge: 'e-badge e-
badge-success', icons: 'promotion', type: 'Primary' },
  { id: 'p_04', text: 'Updates', messages: '13 New', badge: 'e-badge e-
badge-info', icons: 'updates', type: 'Primary' },
  { id: 'p_05', text: 'Starred', messages: '', badge: '', icons:
'starred', type: 'All Labels' },
  { id: 'p_06', text: 'Important', messages: '2 New', badge: 'e-badge e-
badge-danger', icons: 'important', type: 'All Labels' },
  { id: 'p_07', text: 'Sent', messages: '', badge: '', icons: 'sent',
type: 'All Labels' },
  { id: 'p_08', text: 'Outbox', messages: '', badge: '', icons: 'outbox',
type: 'All Labels' },
  { id: 'p_09', text: 'Drafts', messages: '7 New', badge: 'e-badge e-
badge-warning', icons: 'draft', type: 'All Labels' },
];
var list = new ej.lists.ListView({
  // Bind listview datasource
  dataSource: dataSource,
  // Assign header title
  headerTitle: 'Inbox',
  // Enable header
  showHeader: true,
  // Assign template
  template: '<div class="listWrapper" style="width: inherit; height:
inherit;"><span class="${icons} list_svg">&#160;</span>' +
    '<span class="list_text">${text}</span>' +
    '<span class="${badge}" style="float: right; margin-top: 16px; font-
size: 12px;">${messages}</span></div>',
  // Map fields
  fields: { groupBy: 'type' },
  // Bind actioncomplete event
  actionComplete: () => {
    var list =
document.getElementById('lists').getElementsByClassName('e-list-group-
item')[0];
    list.style.display = 'none';
  }
});
```

```
list.appendTo('#lists');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for Badge </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for Badge UI Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element">
      <div class="sample_container badge-list">
        <!-- Listview element -->
        <div id="lists"></div>
      </div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Dynamic badge content in ##Platform_Name## Badge control

Badges in real-time needs to be updated dynamically based on the requirements. The following sample demonstrates how to update the badges content dynamically. Click the increment button to change the badge value.

INDEX.JS

```
// Datasource for listview, badge field is the class data for Badges
var dataSource = [
  { id: 'p_01', text: 'Primary', messages: '3 New', badge: 'e-badge e-
badge-primary', icons: 'primary', type: 'Primary' },
```

```

    { id: 'p_02', text: 'Social', messages: '27 New', badge: 'e-badge e-
    badge-secondary', icons: 'social', type: 'Primary' },
    { id: 'p_03', text: 'Promotions', messages: '7 New', badge: 'e-badge e-
    badge-success', icons: 'promotion', type: 'Primary' },
    { id: 'p_04', text: 'Updates', messages: '13 New', badge: 'e-badge e-
    badge-info', icons: 'updates', type: 'Primary' },
    { id: 'p_05', text: 'Starred', messages: '', badge: '', icons:
    'starred', type: 'All Labels' },
    { id: 'p_06', text: 'Important', messages: '2 New', badge: 'e-badge e-
    badge-danger', icons: 'important', type: 'All Labels' },
    { id: 'p_07', text: 'Sent', messages: '', badge: '', icons: 'sent',
    type: 'All Labels' },
    { id: 'p_08', text: 'Outbox', messages: '', badge: '', icons: 'outbox',
    type: 'All Labels' },
    { id: 'p_09', text: 'Drafts', messages: '7 New', badge: 'e-badge e-
    badge-warning', icons: 'draft', type: 'All Labels' },
  ];
  var list = new ej.lists.ListView({
    // Bind listview datasource
    dataSource: dataSource,
    // Assign header title
    headerTitle: 'Inbox',
    // Enable header
    showHeader: true,
    // Assign template
    template: '<div class="listWrapper" style="width: inherit; height:
    inherit;"><span class="${icons} list_svg">&#160;</span>' +
      '<span class="list_text">${text}</span>' +
      '${if(messages !== "")}<span class="${badge}" style="float:
    right; margin-top: 16px; font-size: 12px;">${messages}</span>${/if}</div>',
    // Map fields
    fields: { groupBy: 'type' },
    // Bind actioncomplete event
    actionComplete: () => {
      badgeElements =
      Array.prototype.slice.call(document.getElementById('lists').getElementsByClassName('e-badge'));
    }
  });
  list.appendTo('#lists');
  document.getElementById('button').addEventListener('click', function
  buttonClick() {
    badgeElements.forEach((element) => {
      element.textContent = (Number(element.textContent.split(' ')[0])) +
      1 + ' New';
    })
  });
});

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for Badge </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for Badge UI Control">
  <meta name="author" content="Syncfusion">

```

```

<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element">
            <div class="sample_container badge-list">
                <!-- Listview element -->
                <div id="lists"></div>
                <p class="crossline"></p>
                <span class="incr_button">
                    <button class="e-btn e-primary" id="button">Increment
Badge Count</button>
                </span>
            </div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Barcode

Getting started in ##Platform_Name## Barcode control

This section explains the steps used to create a simple Barcode and demonstrates the basic usage of the barcode component using Essential JS 2

[quickstart](#) seed repository. This seed repository is pre-configured with the Essential JS 2 package.

Dependencies

Following is the list of minimum dependencies required to use the barcode.

```
|-- @syncfusion/ej2-barcode-generator
```

```
|-- @syncfusion/ej2-base
```

```
|-- @syncfusion/ej2-data
```

Setup for local development

Clone the Essential JS 2 quickstart application project from [GitHub](#), and install the necessary npm packages using the following command line scripts.

```
git clone https://github.com/syncfusion/ej2-quickstart.git quickstart
cd quickstart
npm install
```

Configuring system JS

[Syncfusion BarcodeGenerator packages](#) have to be mapped in the `system.config.js` configuration file.

```
System.config({
  paths: {
    'syncfusion:': './node_modules/@syncfusion/',
  },
  map: {
    app: 'app',
    //Syncfusion packages mapping
    "@syncfusion/ej2-base": "syncfusion:ej2-base/dist/ej2-base.umd.min.js",
    "@syncfusion/ej2-data": "syncfusion:ej2-data/dist/ej2-data.umd.min.js",
    "@syncfusion/ej2-buttons": "syncfusion:ej2-buttons/dist/ej2-buttons.umd.min.js",
    "@syncfusion/ej2-splitbuttons": "syncfusion:ej2-splitbuttons/dist/ej2-splitbuttons.umd.min.js",
    "@syncfusion/ej2-popups": "syncfusion:ej2-popups/dist/ej2-popups.umd.min.js",
    "@syncfusion/ej2-navigations": "syncfusion:ej2-navigations/dist/ej2-navigations.umd.min.js",
    "@syncfusion/ej2-inputs": "syncfusion:ej2-inputs/dist/ej2-inputs.umd.min.js",
    "@syncfusion/ej2-dropdowns": "syncfusion:ej2-dropdowns/dist/ej2-dropdowns.umd.min.js",
    "@syncfusion/ej2-calendars": "syncfusion:ej2-calendars/dist/ej2-calendars.umd.min.js",
    "@syncfusion/ej2-lists": "syncfusion:ej2-lists/dist/ej2-lists.umd.min.js",
    "@syncfusion/ej2-barcode-generator": "syncfusion:ej2-barcode-generator/dist/ej2-barcode-generator.umd.min.js",
    "@syncfusion/ej2-excel-export": "syncfusion:ej2-excel-export/dist/ej2-excel-export.umd.min.js",
    "@syncfusion/ej2-pdf-export": "syncfusion:ej2-pdf-export/dist/ej2-pdf-export.umd.min.js",
    "@syncfusion/ej2-file-utils": "syncfusion:ej2-file-utils/dist/ej2-file-utils.umd.min.js",
    "@syncfusion/ej2-compression": "syncfusion:ej2-compression/dist/ej2-compression.umd.min.js"
```

```
},  
packages: {  
  'app': { main: 'app', defaultExtension: 'js' }  
}  
});  
System.import('app');  
`
```

Adding CSS reference

Combined CSS files are available in the Essential JS 2 package root folder. This can be referenced in your [src/styles/styles.css] using the following code.

```
`  
@import '../node_modules/@syncfusion/ej2/material.css';  
`
```

Adding Barcode Generator control

You can start adding Essential JS 2 barcode-generator component to the application. To get started, add the barcode component in `app.ts` and `index.html` files using the following code.

Place the following barcode-generator code in the `app.ts`.

INDEX.JS

```
var barcode = new ej.barcodegenerator.BarcodeGenerator(  
  {  
    width: '200px',  
    height: '150px',  
    mode: 'SVG',  
    type: 'Codabar',  
    value: '123456789',  
  }  
);  
barcode.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>  
  <title>EJ2 Barcode-Generator</title>  
  <meta charset="utf-8">  
  <meta name="viewport" content="width=device-width, initial-scale=1.0">  
  <meta name="description" content="Typescript UI Controls">  
  <meta name="author" content="Syncfusion">  
  <link href="index.css" rel="stylesheet">  
  
  <style>  
    .barcodeStyle{  
      height: 150px;  
      width: 200px;  
      padding-left: 40%;  
    }  
  </style>  
</head><body>  
  <div id="element">  
    <ej-barcode-generator value="123456789" mode="SVG" type="Codabar">  
    </ej-barcode-generator>  
  </div>  
</body></html>
```



```

        padding-top: 9%;
    }
</style><script
src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" class="barcodeStyle">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Now, add an HTML div element to act as the barcode element in `index.html` using the following code.

```

`html
<!DOCTYPE html>
<html lang="en">
<head>
<title>Essential JS 2</title>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-scale=1.0, user-scalable=no" />
<meta name="description" content="Essential JS 2" />
<meta name="author" content="Syncfusion" />
<link rel="shortcut icon" href="resources/favicon.ico" />
<link href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css" rel="stylesheet"
/>
<!--style reference from app-->
<link href="/styles/styles.css" rel="stylesheet" />
<!--system js reference and configuration-->
</head>
<body>
<!--Element which will render as Barcode-->
<div id="barcode"></div>

```

</body>

</html>

,

Adding QR Generator control

You can add the QR code in our barcode generator component.

INDEX.JS

```
var barcode = new ej.barcodegenerator.QRCodeGenerator
(
  {
    width: '200px',
    height: '150px',
    mode: 'SVG',
    value: 'Syncfusion',
  }
);
barcode.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Barcode-Generator</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

<style>
  .barcodeStyle{
    height: 150px;
    width: 200px;
    padding-left: 40%;
    padding-top: 9%;
  }
</style><script
src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" class="barcodeStyle">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
```

```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Adding Datamatrix Generator control

You can add the datamatrix code in our barcode generator component.

INDEX.JS

```
var barcode = new ej.barcodegenerator.DataMatrixGenerator
(
    {
        width: '200px',
        height: '150px',
        displayText: { visibility: false },
        mode: 'SVG',
        value: 'Syncfusion',
    }
);
barcode.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Barcode-Generator</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

<style>
    .barcodeStyle{
        height: 150px;
        width: 200px;
        padding-left: 40%;
        padding-top: 9%;
    }
</style><script
src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" class="barcodeStyle">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
```

```
</body></html>
```

BarcodeGenerator in ##Platform_Name## Barcode control

Code39

The Code 39 character set includes the digits 0-9, the letters A-Z (upper case only), and the symbols: space, minus (-), plus (+), period (.), dollar sign (\$), slash (/), and percent (%). A special start / stop character is placed at the beginning and ending of each barcode. The barcode can be of any length; even more than 25 characters begin to push the bounds. Code 39 is the only type of barcode that does not require a checksum for common use.

INDEX.JS

```
var barcode = new ej.barcodegenerator.BarcodeGenerator(  
    {  
        width: '200px',  
        height: '150px',  
        mode: 'SVG',  
        type: 'Code39',  
        value: 'SYNCFUSION',  
    }  
);  
barcode.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>  
    <title>EJ2 Barcode</title>  
    <meta charset="utf-8">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <meta name="description" content="Typescript UI Controls">  
    <meta name="author" content="Syncfusion">  
    <link href="index.css" rel="stylesheet">  
  
    <style>  
        .barcodeStyle{  
            height: 150px;  
            width: 200px;  
            padding-left: 40%;  
            padding-top: 9%;  
        }  
    </style><script  
src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"  
type="text/javascript"></script>  
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type  
="text/javascript"></script>  
</head>  
<body>  
  
    <div id="container" class="barcodeStyle">  
        <div id="element"></div>  
    </div>  
<script>  
var ele = document.getElementById('container');
```

```

if(ele) {
    ele.style.visibility = "visible";
}

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Code39 Extended

Code 39 Extended is an extended version of Code 39 that supports ASCII character set. In Code 39 Extended, you can also code 26 lower letters (a-z) and the special characters in the keyboard.

INDEX.JS

```

var barcode = new ej.barcodegenerator.BarcodeGenerator(
{
    width: '200px',
    height: '150px',
    mode: 'SVG',
    type: 'Code39Extension',
    value: 'SYNCFUSION',
}
);
barcode.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Barcode</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

<style>
    .barcodeStyle{
        height: 150px;
        width: 200px;
        padding-left: 40%;
        padding-top: 9%;
    }
</style><script
src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" class="barcodeStyle">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');

```

```

if(ele) {
    ele.style.visibility = "visible";
}

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Code 11

Code 11 is used primarily for labeling the telecommunication equipment's. The character set includes the digits 0 to 9, a dash (-), and a start / stop code.

INDEX.JS

```

var barcode = new ej.barcodegenerator.BarcodeGenerator(
{
    width: '200px',
    height: '150px',
    mode: 'SVG',
    type: 'Code11',
    value: '112',
}
);
barcode.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Barcode</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

<style>
    .barcodeStyle{
        height: 150px;
        width: 200px;
        padding-left: 40%;
        padding-top: 9%;
    }
</style><script
src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" class="barcodeStyle">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');

```

```

if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Codabar

Codabar is a variable length symbol that encodes the following 20 characters:

0123456789-\$.+ABCD

The characters, A, B, C and D are used as start and stop characters. Codabar is used in libraries, blood banks, the package delivery industry and a variety of other information processing applications.

INDEX.JS

```

var barcode = new ej.barcodegenerator.BarcodeGenerator(
{
    width: '200px',
    height: '150px',
    mode: 'SVG',
    type: 'Codabar',
    value: '123456789',
}
);
barcode.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Barcode</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

<style>
    .barcodeStyle{
        height: 150px;
        width: 200px;
        padding-left: 40%;
        padding-top: 9%;
    }
</style><script
src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" class="barcodeStyle">
        <div id="element"></div>

```

```

    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Code 32

Code 32 is mainly used for coding pharmaceuticals, cosmetics and dietetics. It is often used to encode Italian Pharmacode that has the following structure:

- 'A' character (ASCII 65), that is not really encoded.
- 8 digits for Pharmacode (It generally begins with / and prefixed with 0).
- 1 digit for checksum module 10, that is automatically calculated by barcode.

The value to be encoded must be 8 digits Pharmacode (prefix it with '0' if necessary) and the 9th digit (the checksum) is automatically calculated by barcode.

INDEX.JS

```

var barcode = new ej.barcodegenerator.BarcodeGenerator(
{
    type: 'Code32',
    value: '01234567',
    width: '200px', height: '150px',
    mode: 'SVG',
}
);
barcode.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Barcode</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

<style>
    .barcodeStyle{
        height: 150px;
        width: 200px;
        padding-left: 40%;
        padding-top: 9%;
    }
</style><script
src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>

```



```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" class="barcodeStyle">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Code 93

Code 93 is designed to complement and improve upon Code 39. It can represent the entire ASCII character set by using combinations of 2 characters. Code 93 is a continuous, variable-length symbology and produces denser code. The Standard Mode (default implementation) can encode uppercase letters (A-Z), digits (0-9), and special characters like *, -, \$, %, (Space), ., /, and +.

INDEX.JS

```

var barcode = new ej.barcodegenerator.BarcodeGenerator(
{
    type: 'Code93',
    value: '01234567',
    width: '200px', height: '150px',
    mode: 'SVG',
}
);
barcode.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Barcode</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

<style>
    .barcodeStyle{
        height: 150px;
        width: 200px;
        padding-left: 40%;
        padding-top: 9%;
    }

```

```

</style><script
src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" class="barcodeStyle">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Code 93 Extended

The Code 93 Extended Barcode symbology is continuous, variable length, and self-checking. It is based on Code 93 Barcode. The Extended Version can encode all 128 ASCII characters.

Code 128

Code 128 is a variable length, high density, alphanumeric, linear bar code symbology, capable of encoding the full 128-character ASCII character set and extended character sets. This symbology includes a checksum digit for verification and the barcode can also be verified character-by-character by verifying the parity of each data byte.

Code 128 Code Sets

- Code Set A (or Chars Set A) includes all of the standard upper case U.S. alphanumeric keyboard characters and punctuation characters along with the control characters, (namely, characters with ASCII values from 0 to 95 inclusive), and seven special characters.
- Code Set B (or Chars Set B) includes all of the standard upper case alphanumeric keyboard characters and punctuation characters along with the lower case alphabetic characters (namely, characters with ASCII values from 32 to 127 inclusive), and seven special characters.
- Code Set C (or Chars Set C) includes the set of 100 digit pairs from 00 to 99 inclusive along with three special characters. This allows numeric data to be encoded as two data digits per symbol character, at effectively twice the density of standard data.

Code 128 Special characters

The last seven characters of Code Sets A and B (character values 96 - 102) and the last three characters of Code Set C (character values 100 - 102) are special non-data characters with no ASCII character equivalents that have a particular significance to the Barcode reading device.

INDEX.JS

```

var barcode = new ej.barcodegenerator.BarcodeGenerator (
{
    type: 'Code128',
    value: 'SYNCFUSION',

```

```

        width: '200px', height: '150px',
        mode: 'SVG',
    }
);
barcode.appendTo('#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Barcode</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <style>
    .barcodeStyle{
      height: 150px;
      width: 200px;
      padding-left: 40%;
      padding-top: 9%;
    }
  </style><script
src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" class="barcodeStyle">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Customizing the Barcode color

A page or printed media with barcode often appears colorful in the background and surrounding region with other contents. In such cases the barcode can also be customized to suit the needs. You can achieve this by using for forecolor property .

INDEX.JS

```

var barcode = new ej.barcodegenerator.BarcodeGenerator(
{
  type: 'Code128',
  value: 'SYNCFUSION',
```

```

        width: '200px', height: '150px',
        mode: 'SVG',
        foreColor: 'red',
    }
);
barcode.appendTo('#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Barcode</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <style>
    .barcodeStyle{
      height: 150px;
      width: 200px;
      padding-left: 40%;
      padding-top: 9%;
    }
  </style><script
src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" class="barcodeStyle">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Customizing the Barcode dimension

The dimension of the barcode can be changed using the height and width property of the barcodegenerator.

INDEX.JS

```

var barcode = new ej.barcodegenerator.BarcodeGenerator (
{
  type: 'Code128',
  value: 'SYNCFUSION',
```

```

        width: '300px', height: '300px',
        mode: 'SVG',
    }
);
barcode.appendTo('#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Barcode</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <style>
    .barcodeStyle{
      height: 150px;
      width: 200px;
      padding-left: 40%;
      padding-top: 9%;
    }
  </style><script
src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" class="barcodeStyle">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Customizing the text

In barcode generators you can customize the barcode text by using display text property .

INDEX.JS

```

var barcode = new ej.barcodegenerator.BarcodeGenerator(
{
  type: 'Code128',
  value: 'SYNCFUSION',
  width: '200px', height: '150px',
  mode: 'SVG',
  displayText: {text: 'text'},
```

```

    }
);
barcode.appendTo('#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Barcode</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

<style>
  .barcodeStyle{
    height: 150px;
    width: 200px;
    padding-left: 40%;
    padding-top: 9%;
  }
</style><script
src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" class="barcodeStyle">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Qrcodegenerator in ##Platform_Name## Barcode control

QR Code

A QR Code is a two-dimensional barcode that consists of a grid of dark and light dots or blocks that form a square. The data encoded in the barcode can be numeric, alphanumeric, or Shift Japanese Industrial Standards (JIS8) characters. The QR Code uses version from 1 to 40. Version 1 measures 21 modules x 21 modules, Version 2 measures 25 modules x 25 modules, and so on. The number of modules increases in steps of 4 modules per side up to Version 40 that measures 177 modules x 177 modules. Each version has its own capacity. By default, the barcode control automatically set the version according to the length of the input text. The QR Barcodes are designed for industrial uses and also commonly used in consumer advertising.

INDEX.JS

```
var barcode = new ej.barcodegenerator.QRCodeGenerator
(
    {
        width: '200px',
        height: '200px',
        displayText: { visibility: false },
        mode: 'SVG',
        value: 'Syncfusion',
    }
);
barcode.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Barcode</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

<style>
    .barcodeStyle{
        height: 150px;
        width: 200px;
        padding-left: 40%;
        padding-top: 9%;
    }
</style><script
src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" class="barcodeStyle">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Customizing the Barcode color

A page or printed media with barcode often appears colorful in the background and surrounding region with other contents. In such cases the barcode can also be customized to suit the needs. You can achieve this by using for forecolor property .

INDEX.JS

```
var barcode = new ej.barcodegenerator.QRCodeGenerator
(
  {
    width: '200px',
    height: '200px',
    displayText: { visibility: false },
    mode: 'SVG',
    value: 'Syncfusion',
    foreColor: 'red',
  }
);
barcode.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Barcode</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <style>
    .barcodeStyle{
      height: 150px;
      width: 200px;
      padding-left: 40%;
      padding-top: 9%;
    }
  </style><script
src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" class="barcodeStyle">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
<script src="index.js" type="text/javascript"></script>
```



```
</body></html>
```

Customizing the Barcode dimension

The dimension of the barcode can be changed using the height and width properties of the barcodegenerator.

INDEX.JS

```
var barcode = new ej.barcodegenerator.QRCodeGenerator
(
  {
    width: '100px',
    height: '100px',
    displayText: { visibility: false },
    mode: 'SVG',
    value: 'Syncfusion',
  }
);
barcode.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Barcode</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

<style>
  .barcodeStyle{
    height: 150px;
    width: 200px;
    padding-left: 40%;
    padding-top: 9%;
  }
</style><script
src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" class="barcodeStyle">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
```

```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Customizing the text

In barcode generators You can customize the barcode text by using display text property .

INDEX.JS

```
var barcode = new ej.barcodegenerator.QRCodeGenerator
(
    {
        width: '200px',
        height: '200px',
        displayText: { visibility: true },
        mode: 'SVG',
        value: 'Syncfusion',
    }
);
barcode.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Barcode</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

<style>
    .barcodeStyle{
        height: 150px;
        width: 200px;
        padding-left: 40%;
        padding-top: 9%;
    }
</style><script
src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" class="barcodeStyle">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
```

```
</body></html>
```

Datamatrixgenerator in ##Platform_Name## Barcode control

Data Matrix

DataMatrix Barcode is a two dimensional barcode that consists of a grid of dark and light dots or blocks forming square or rectangular symbol. The data encoded in the barcode can either be numbers or alphanumeric. They are widely used in printed media such as labels and letters. You can read it easily with the help of a barcode reader and mobile phones.

INDEX.JS

```
var barcode = new ej.barcodegenerator.DataMatrixGenerator
(
    {
        width: '200px',
        height: '150px',
        displayText: { visibility: false },
        mode: 'SVG',
        value: 'Syncfusion',
    }
);
barcode.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Barcode</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <style>
        .barcodeStyle{
            height: 150px;
            width: 200px;
            padding-left: 40%;
            padding-top: 9%;
        }
    </style><script
src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" class="barcodeStyle">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
```

```

if(ele) {
    ele.style.visibility = "visible";
}

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customizing the Barcode color

A page or printed media with barcode often appears colorful in the background and surrounding region with other contents. In such cases the barcode can also be customized to suit the needs. You can achieve this by using the forecolor property .

INDEX.JS

```

var barcode = new ej.barcodegenerator.DataMatrixGenerator
(
    {
        width: '200px',
        height: '200px',
        displayText: { visibility: false },
        mode: 'SVG',
        value: 'Syncfusion',
        foreColor: 'red',
    }
);
barcode.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Barcode</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <style>
        .barcodeStyle{
            height: 150px;
            width: 200px;
            padding-left: 40%;
            padding-top: 9%;
        }
    </style><script
src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" class="barcodeStyle">
        <div id="element"></div>

```

```
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Customizing the Barcode dimension

The dimension of the barcode can be changed using the height and width property of the barcodegenerator.

INDEX.JS

```
var barcode = new ej.barcodegenerator.DataMatrixGenerator
(
    {
        width: '100px',
        height: '100px',
        displayText: { visibility: false },
        mode: 'SVG',
        value: 'Syncfusion',
    }
);
barcode.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Barcode</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <style>
        .barcodeStyle{
            height: 150px;
            width: 200px;
            padding-left: 40%;
            padding-top: 9%;
        }
    </style><script
src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" class="barcodeStyle">
```

```

        <div id="element"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customizing the text

In barcode generators you can customize the barcode text by using the display text property .

INDEX.JS

```

var barcode = new ej.barcodegenerator.DataMatrixGenerator
(
    {
        width: '200px',
        height: '200px',
        displayText: { visibility: true },
        mode: 'SVG',
        value: 'Syncfusion',
    }
);
barcode.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Barcode</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

</head>
<body>
    <div id="container" class="barcodeStyle">
        <div id="element"></div>
    </div>
</body>
</html>

```

```

    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Export in ##Platform_Name## Barcode control

Export barcode as an image and base64 string is common for barcode,QRcode and datamatrix. The following code samples explain how to export barcode as an image and base64 string.

Export

Barcode provides the support to export its content as an image in the specified image type and downloads it in the browser.

The following code example shows how to export the barcode as an image

`ts

```

import { BarcodeGenerator, BarcodeExportType } from '@syncfusion/ej2-barcode-generator';
let barcode: BarcodeGenerator = new BarcodeGenerator({
width: '200px', height: '150px',
type: 'Code39',
value: 'BARCODE',
displayText: { text: 'ABCD' },
});
barcode.appendTo('#element');
let filename: string = 'Export';
barcode.exportImage(filename,'JPG');
`

```

The filename specifies the name of the file to be downloaded.

Export As Base64String

Barcode provides the support to export its content as an image in the specified image type and returns it as base64 string.

The following code example shows how to export the barcode as a base64 string

`ts

```

import { BarcodeGenerator, BarcodeExportType } from '@syncfusion/ej2-barcode-generator';
let barcode: BarcodeGenerator = new BarcodeGenerator({
width: '200px', height: '150px',
type: 'Code39',

```

```

value: 'BARCODE',
displayText: { text: 'ABCD' },
});
barcode.appendTo('#element');
async function () {
// Can able to store the return base64 string in variable
var data = await barcode.exportAsBase64Image('JPG');
};
`

```

Note:

Format is to specify the type or format of the exported file. You can export the barcode to the following formats:

* JPG.

* PNG.

Breadcrumb

Data binding in ##Platform_Name## Breadcrumb control

The Breadcrumb supports to generate items based on the current URL by default. You can set the [items](#) property or [url](#) property to generate the items.

Items based on current Url

The breadcrumb items can be generated based on the current URL of the page when the user does not specify the breadcrumb items using [items](#) property. The following example shows the breadcrumb items generated from the provided URL in the component.

INDEX.JS

```

ej.base.enableRipple(true);
new ej.navigations.Breadcrumb({
  enableNavigation: false
}, '#breadcrumb');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <!--style reference from app-->

```



```

<link href="styles.css" rel="stylesheet">
<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="control-section">
            <nav id="breadcrumb"></nav>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

This sample is hosted in different location, so the Breadcrumb component is rendered with different location instead of the actual location.

Absolute Url

The breadcrumb items can be generated based on the [url](#) property in the component when the user does not specify the breadcrumb items using [items](#) property. The following example shows the breadcrumb items generated from the provided url in the component.

INDEX.JS

```

ej.base.enableRipple(true);
new ej.navigations.Breadcrumb({
    enableNavigation: false,
    url: "https://ej2.syncfusion.com/demos/breadcrumb/bind-to-location"
}, '#breadcrumb');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Essential JS 2">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <!--style reference from app-->

```

```

<link href="styles.css" rel="stylesheet">
<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="control-section">
            <nav id="breadcrumb"></nav>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customize text when generated items using Url

The breadcrumb items text can be customized by using the [beforeItemRender](#) event. In the following example, **bind-to-location** text was changed as **location**.

INDEX.JS

```

ej.base.enableRipple(true);
new ej.navigations.Breadcrumb({
    enableNavigation: false,
    url: "https://ej2.syncfusion.com/demos/breadcrumb/bind-to-location",
    beforeItemRender: function(args) {
        if (args.item.text === 'bind-to-location') {
            args.item.text = 'location';
        }
    }
}, '#breadcrumb');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Essential JS 2">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">

```

```

<!--style reference from app-->
<link href="styles.css" rel="stylesheet">
<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="control-section">
            <nav id="breadcrumb"></nav>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Icons in ##Platform_Name## Breadcrumb control

The Breadcrumb component contains an icon/image to provide a visual representation of an item.

Icon in Breadcrumb item

To load the icon/image on the breadcrumb item, set the [iconCss](#) property.

Breadcrumb with Font Icon

To place the font icon on the breadcrumb item, set the [iconCss](#) property to **e-icons** with the required icon CSS. By default, the icon is positioned to the left side of the item.

INDEX.JS

```

ej.base.enableRipple(true);
var items = [
    {
        iconCss: 'e-icons e-home',
        url: "https://ej2.syncfusion.com/demos",
    },
    {
        text: "Components",
        url: "https://ej2.syncfusion.com/demos/#/material/grid/grid-
overview",
    },
    {
        text: "Navigations",
        url:
"https://ej2.syncfusion.com/demos/#/material/breadcrumb/default",
    },
    {
        text: "Breadcrumb",

```

```

        url: "../breadcrumb/default",
    }
];
new ej.navigations.Breadcrumb({
    items: items,
    enableNavigation: false
}, '#breadcrumb');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Essential JS 2">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <!--style reference from app-->
    <link href="styles.css" rel="stylesheet">
    <!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="control-section">
            <nav id="breadcrumb"></nav>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Breadcrumb with Image

In the Breadcrumb component, images can be added for the items using the [iconCss](#) property. In the following example, an image was added to the breadcrumb item by using the iconCss class `e-image-home` and specifying height and width for the css class.

INDEX.JS

```
ej.base.enableRipple(true);
```

```

var items = [
    {
        iconCss: "e-image-home",
        url: "https://ej2.syncfusion.com/demos",
    },
    {
        text: "Components",
        url: "https://ej2.syncfusion.com/demos/#/material/grid/grid-
overview",
    },
    {
        text: "Navigations",
        url:
"https://ej2.syncfusion.com/demos/#/material/breadcrumb/default",
    },
    {
        text: "Breadcrumb",
        url: "./breadcrumb/default",
    }
];
new ej.navigations.Breadcrumb({
    items: items,
    enableNavigation: false
}, '#breadcrumb');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Essential JS 2">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <!--style reference from app-->
    <link href="styles.css" rel="stylesheet">
    <!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="control-section">
            <nav id="breadcrumb"></nav>
        </div>
    </div>
</script>

```

```

var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Breadcrumb with SVG Image

In the Breadcrumb component, SVG image can be added for the items using the [iconCss](#) property. In the following example, SVG image was added to the breadcrumb item by using the iconCss class `e-svg-home` and specifying height and width for the css class.

INDEX.JS

```

ej.base.enableRipple(true);
var items = [
    {
        iconCss: "e-svg-home",
        url: "https://ej2.syncfusion.com/demos",
    },
    {
        text: "Components",
        url: "https://ej2.syncfusion.com/demos/#/material/grid/grid-overview",
    },
    {
        text: "Navigations",
        url: "https://ej2.syncfusion.com/demos/#/material/breadcrumb/default",
    },
    {
        text: "Breadcrumb",
        url: "../breadcrumb/default",
    }
];
new ej.navigations.Breadcrumb({
    items: items,
    enableNavigation: false
}, '#breadcrumb');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0, user-scalable=no">
    <meta name="description" content="Essential JS 2">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">
    <!--style reference from app-->

```

```

<link href="styles.css" rel="stylesheet">
<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="control-section">
            <nav id="breadcrumb"></nav>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Icon Position

INDEX.JS

```

ej.base.enableRipple(true);
var items = [
    {
        text: "Program Files",
        iconCss: "e-icons e-folder"
    },
    {
        text: "Services",
        iconCss: "e-icons e-folder"
    },
    {
        text: "Config.json",
        iconCss: "e-icons e-file"
    }
];
new ej.navigations.Breadcrumb({
    items: items,
    enableNavigation: false
}, '#breadcrumb');
new ej.navigations.Breadcrumb({
    items: items,
    enableNavigation: false,
    beforeItemRender: function(args) {
        if(args.item.text !== '') {
            args.element.classList.add('e-icon-right');
        }
    }
}, '#breadcrumb2');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="breadcrumb-control" class="control-section">
      <div class="header"><b>Icon Position - Left</b></div><br>
      <nav id="breadcrumb"></nav>
      <br><br>
      <div class="header"><b>Icon Position - Right</b></div><br>
      <nav id="breadcrumb2"></nav>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Icon Only

To display only icons for the items, add icons using the [iconCss](#) property. In the following example, breadcrumb items were demonstrated with only icons by providing the [iconCss](#) property.

INDEX.JS

```

ej.base.enableRipple(true);
var items = [
  {
    iconCss: 'e-icons e-home'
  },

```



```

    {
        iconCss: "e-icons e-folder"
    },
    {
        iconCss: "e-icons e-file"
    }
];
new ej.navigations.Breadcrumb({
    items: items,
    enableNavigation: false
}, '#breadcrumb');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="control-section">
      <nav id="breadcrumb"></nav>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Show icon only for first item

To show icon only for the first item in the Breadcrumb component, add icons to the first item using the [iconCss](#) property. In the following example, the icon was provided only for the first item by setting the [iconCss](#) property.

INDEX.JS

```

ej.base.enableRipple(true);
var items = [
    {
        iconCss: "e-svg-home",
        url: "https://ej2.syncfusion.com/demos",
    },
    {
        text: "Components",
        url: "https://ej2.syncfusion.com/demos/#/material/grid/grid-
overview",
    },
    {
        text: "Navigations",
        url:
"https://ej2.syncfusion.com/demos/#/material/breadcrumb/default",
    },
    {
        text: "Breadcrumb",
        url: "../breadcrumb/default",
    }
];
new ej.navigations.Breadcrumb({
    items: items,
    enableNavigation: false
}, '#breadcrumb');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Essential JS 2">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <!--style reference from app-->
    <link href="styles.css" rel="stylesheet">
    <!--system js reference and configuration-->

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="control-section">
            <nav id="breadcrumb"></nav>

```

```
        </div>
    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Navigation in ##Platform_Name## Breadcrumb control

Breadcrumb navigations support you to provide relative or absolute URL for breadcrumb items, enable navigation for the last item of the Breadcrumb component, and open URL in a new tab or new page.

URL

In the Breadcrumb component, the item represents the URL. The breadcrumb items can be provided with either relative or absolute URL.

Relative URL

INDEX.JS

```
ej.base.enableRipple(true);
var items = [
    {
        text: "Home",
        url: "../"
    },
    {
        text: "Getting",
        url: "../breadcrumb/getting-started"
    },
    {
        text: "Icons",
        url: "../breadcrumb/icons"
    },
    {
        text: "Navigation",
        url: "../breadcrumb/navigation"
    },
    {
        text: "Overflow",
        url: "../breadcrumb/overflow"
    }
];
new ej.navigations.Breadcrumb({
    items: items,
    enableNavigation: false
}, '#breadcrumb');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2</title>
    <meta charset="utf-8">
```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
<meta name="description" content="Essential JS 2">
<meta name="author" content="Syncfusion">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<!--style reference from app-->
<link href="styles.css" rel="stylesheet">
<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="control-section">
            <nav id="breadcrumb"></nav>
        </div>
    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Absolute URL

INDEX.JS

```

ej.base.enableRipple(true);
var items = [
    {
        text: "Home",
        url: "https://ej2.syncfusion.com/documentation/introduction/"
    },
    {
        text: "Getting",
        url:
"https://ej2.syncfusion.com/documentation/breadcrumb/getting-started"
    },
    {
        text: "Icons",
        url: "https://ej2.syncfusion.com/documentation/breadcrumb/icons"
    },
    {
        text: "Navigation",
        url:
"https://ej2.syncfusion.com/documentation/breadcrumb/navigation"
    }
]

```

```

    },
    {
        text: "Overflow",
        url:
            "https://ej2.syncfusion.com/documentation/breadcrumb/overflow"
    }
];
new ej.navigations.Breadcrumb({
    items: items,
    enableNavigation: false
}, '#breadcrumb');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="control-section">
      <nav id="breadcrumb"></nav>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Enable navigation for last Breadcrumb item

Breadcrumb enables the navigation for the last item by setting the [enableActiveItemNavigation](#) property to true. In the following example, the last item of the Breadcrumb was enabled.

INDEX.JS

```

ej.base.enableRipple(true);
var items = [
    {
        text: "Home",
        url: "https://ej2.syncfusion.com/documentation/introduction/"
    },
    {
        text: "Getting",
        url:
"https://ej2.syncfusion.com/documentation/breadcrumb/getting-started"
    },
    {
        text: "Icons",
        url: "https://ej2.syncfusion.com/documentation/breadcrumb/icons"
    },
    {
        text: "Navigation",
        url:
"https://ej2.syncfusion.com/documentation/breadcrumb/navigation"
    },
    {
        text: "Overflow",
        url:
"https://ej2.syncfusion.com/documentation/breadcrumb/overflow"
    }
];
new ej.navigations.Breadcrumb({
    items: items,
    enableNavigation: false,
    enableActiveItemNavigation: true
}, '#breadcrumb');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Essential JS 2">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <!--style reference from app-->
    <link href="styles.css" rel="stylesheet">
    <!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>

```

```
</head>
<body>

  <div id="container">
    <div class="control-section">
      <nav id="breadcrumb"></nav>
    </div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Open URL in new page or tab

To open the url provided in the items in a new page or tab, set the target property of the anchor element for the required item to "_blank" in the [beforeItemRender](#) event of Breadcrumb component. In the following example, the target property of anchor element for the item was set to "_blank" by using the [beforeItemRender](#) event which locates to the path in the new tab.

INDEX.JS

```
ej.base.enableRipple(true);
var items = [
  {
    text: "Home",
    url: "https://ej2.syncfusion.com/documentation/introduction/"
  },
  {
    text: "Getting",
    url:
"https://ej2.syncfusion.com/documentation/breadcrumb/getting-started"
  },
  {
    text: "Icons",
    url: "https://ej2.syncfusion.com/documentation/breadcrumb/icons"
  },
  {
    text: "Navigation",
    url:
"https://ej2.syncfusion.com/documentation/breadcrumb/navigation"
  },
  {
    text: "Overflow",
    url:
"https://ej2.syncfusion.com/documentation/breadcrumb/overflow"
  }
];
new ej.navigations.Breadcrumb({
  items: items,
  beforeItemRender: function(args) {
    if (args.element.children[0]) {
      args.element.children[0].target = "_blank";
    }
  }
});
```

```

    }
  }, '#breadcrumb');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="control-section">
      <nav id="breadcrumb"></nav>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Templates in ##Platform_Name## Breadcrumb control

The Breadcrumb component provides a way to customize the items using [itemTemplate](#) and the separators using [separatorTemplate](#) properties.

Item Template

In the following example, Shopping Cart details are used as breadcrumb Items and the each items is rendered as chips component using [itemTemplate](#).

INDEX.JS

```

ej.base.enableRipple(true);
var items = [

```



```

    {
        text: 'Cart'
    },
    {
        text: 'Billing'
    },
    {
        text: 'Shipping'
    },
    {
        text: 'Payment'
    }
];
new ej.navigations.Breadcrumb({
    items: items,
    enableNavigation: false,
    itemTemplate: '<div id="chip-default" class="e-lib e-chip-list e-control e-chip-set" role="listbox" aria-multiselectable="false"><div class="e-chip e-primary" tabindex="0" role="option" aria-label="Apple" aria-selected="false"><span class="e-chip-text">${text}</span></div></div>'
}, '#breadcrumb');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0, user-scalable=no">
    <meta name="description" content="Essential JS 2">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
    <!--style reference from app-->
    <link href="styles.css" rel="stylesheet">
    <!--system js reference and configuration-->

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="control-section">
            <nav id="breadcrumb"></nav>
        </div>
    </div>
<script>
var ele = document.getElementById('container');

```

```

if(ele) {
    ele.style.visibility = "visible";
}

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Separator Template

In the following example, the separators are customized with icons using [separatorTemplate](#). While rendering the separator template, you can get the previous and next item using `previousItem` and `nextItem` variables, respectively.

INDEX.JS

```

ej.base.enableRipple(true);
var items = [
    {
        text: 'Cart'
    },
    {
        text: 'Billing'
    },
    {
        text: 'Shipping'
    },
    {
        text: 'Payment'
    }
];
new ej.navigations.Breadcrumb({
    items: items,
    enableNavigation: false,
    separatorTemplate: '<span class="e-icons e-bullet-arrow"></span>'
}, '#breadcrumb');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Essential JS 2">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <!--style reference from app-->
    <link href="styles.css" rel="stylesheet">
    <!--system js reference and configuration-->

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="control-section">
            <nav id="breadcrumb"></nav>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customize Specific Item Template

The specific breadcrumb item can be customizable using itemTemplate with conditional rendering. In the following example, added the span element only for the **breadcrumb** text in breadcrumb item.

INDEX.JS

```

ej.base.enableRipple(true);
var specificTemplateItems = [
    {
        text: "Home",
        url: "https://ej2.syncfusion.com/demos",
    },
    {
        text: "Components",
        url: "https://ej2.syncfusion.com/demos/#/material/grid/grid-
overview",
    },
    {
        text: "Navigations",
        url: "https://ej2.syncfusion.com/demos/#/material/menu/default",
    },
    {
        text: "Breadcrumb",
        url: "./breadcrumb/default",
    }
];
new ej.navigations.Breadcrumb({
    items: specificTemplateItems,
    itemTemplate: '<div>${if(text=="Breadcrumb")}<span class="e-searchfor-
text"><span style="margin-right: 5px">Search for:</span><a class="e-
breadcrumb-text" href="${url}" onclick="return
false">${text}</a></span>${else}<a class="e-breadcrumb-text" href="${url}"
onclick="return false">${text}</a>${/if}</div>',
    cssClass: 'e-specific-item-template'
}, '#breadcrumb');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="control-section">
      <nav id="breadcrumb"></nav>
    </div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Overflow in ##Platform_Name## Breadcrumb control

In the Breadcrumb component, [maxItems](#) and [overflowMode](#) properties were used to limit the number of breadcrumb items to be displayed.

In the following example, the maxItems is set as 3 with overflowMode as Default. To prevent breadcrumb item navigation, the [enableNavigation](#) property has been set to false in the Breadcrumb component.

INDEX.JS

```

ej.base.enableRipple(true);
var items = [
  {

```

```

        text: "Home",
        url: "../"
    },
    {
        text: "Getting",
        url: "../breadcrumb/getting-started"
    },
    {
        text: "Data-Binding",
        url: "../breadcrumb/data-binding"
    },
    {
        text: "Icons",
        url: "../breadcrumb/icons"
    },
    {
        text: "Navigation",
        url: "../breadcrumb/navigation"
    },
    {
        text: "templates",
        url: "../breadcrumb/templates"
    },
    {
        text: "Overflow",
        url: "../breadcrumb/overflow"
    }
];
new ej.navigations.Breadcrumb({
    items: items,
    enableNavigation: false,
    maxItems: 3,
    separatorTemplate: '<span class="e-icons e-arrow"></span>'
}, '#breadcrumb');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
  user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  navigations/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
```

```
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="control-section">
            <nav id="breadcrumb"></nav>
        </div>
    </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

The following overflow modes are available in the Breadcrumb component.

- Collapsed
- Menu
- Wrap
- Scroll
- Hidden
- None

Collapsed

Collapsed mode shows the first and last Breadcrumb items and hides the remaining items with a collapsed icon. When the collapsed icon is clicked, all items become visible and navigable.

INDEX.JS

```
ej.base.enableRipple(true);
var items = [
    {
        text: "Home",
        url: "../"
    },
    {
        text: "Getting",
        url: "../breadcrumb/getting-started"
    },
    {
        text: "Data-Binding",
        url: "../breadcrumb/data-binding"
    },
    {
        text: "Icons",
        url: "../breadcrumb/icons"
    },
    {
        text: "Navigation",
```

```

        url: "./breadcrumb/navigation"
    },
    {
        text: "templates",
        url: "./breadcrumb/templates"
    },
    {
        text: "Overflow",
        url: "./breadcrumb/overflow"
    }
];
new ej.navigations.Breadcrumb({
    items: items,
    enableNavigation: false,
    maxItems: 3,
    overflowMode: 'Collapsed'
}, '#breadcrumb');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Essential JS 2">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <!--style reference from app-->
    <link href="styles.css" rel="stylesheet">
    <!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="control-section">
            <nav id="breadcrumb"></nav>
        </div>
    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Menu

Menu mode shows the number of Breadcrumb items that can be accommodated within the container space and creates a submenu with the remaining items.

INDEX.JS

```
ej.base.enableRipple(true);
var items = [
    {
        text: "Home",
        url: "../"
    },
    {
        text: "Getting",
        url: "./breadcrumb/getting-started"
    },
    {
        text: "Data-Binding",
        url: "./breadcrumb/data-binding"
    },
    {
        text: "Icons",
        url: "./breadcrumb/icons"
    },
    {
        text: "Navigation",
        url: "./breadcrumb/navigation"
    },
    {
        text: "templates",
        url: "./breadcrumb/templates"
    },
    {
        text: "Overflow",
        url: "./breadcrumb/overflow"
    }
];
new ej.navigations.Breadcrumb({
    items: items,
    enableNavigation: false,
    maxItems: 3,
    overflowMode: 'Menu'
}, '#breadcrumb');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
  user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<!--style reference from app-->
<link href="styles.css" rel="stylesheet">
<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="control-section">
            <nav id="breadcrumb"></nav>
        </div>
    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Wrap

Wrap mode wraps the items to multiple lines when the Breadcrumb's width exceeds the container space.

INDEX.JS

```

ej.base.enableRipple(true);
var items = [
    {
        text: "Home",
        url: "../"
    },
    {
        text: "Getting",
        url: "../breadcrumb/getting-started"
    },
    {
        text: "Data-Binding",
        url: "../breadcrumb/data-binding"
    },
    {
        text: "Icons",
        url: "../breadcrumb/icons"
    },
    {
        text: "Navigation",

```

```

        url: "../breadcrumb/navigation"
    },
    {
        text: "templates",
        url: "../breadcrumb/templates"
    },
    {
        text: "Overflow",
        url: "../breadcrumb/overflow"
    }
];
new ej.navigations.Breadcrumb({
    items: items,
    enableNavigation: false,
    maxItems: 3,
    overflowMode: 'Wrap'
}, '#breadcrumb');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Essential JS 2">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <!--style reference from app-->
    <link href="styles.css" rel="stylesheet">
    <!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="control-section">
            <nav id="breadcrumb"></nav>
        </div>
    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Scroll

Scroll mode shows an HTML scroll bar when the Breadcrumb's width exceeds the container space.

INDEX.JS

```
ej.base.enableRipple(true);
var items = [
    {
        text: "Home",
        url: "../"
    },
    {
        text: "Getting",
        url: "./breadcrumb/getting-started"
    },
    {
        text: "Data-Binding",
        url: "./breadcrumb/data-binding"
    },
    {
        text: "Icons",
        url: "./breadcrumb/icons"
    },
    {
        text: "Navigation",
        url: "./breadcrumb/navigation"
    },
    {
        text: "templates",
        url: "./breadcrumb/templates"
    },
    {
        text: "Overflow",
        url: "./breadcrumb/overflow"
    }
];
new ej.navigations.Breadcrumb({
    items: items,
    enableNavigation: false,
    maxItems: 3,
    overflowMode: 'Scroll',
    separatorTemplate: '<span class="e-icons e-arrow"></span>',
    '#breadcrumb');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Essential JS 2">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
```

```
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<!--style reference from app-->
<link href="styles.css" rel="stylesheet">
<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="control-section">
            <nav id="breadcrumb"></nav>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Hidden

Hidden mode shows the maximum number of items possible in the container space and hides the remaining items. Clicking on a previous item will make the hidden item visible.

INDEX.JS

```
ej.base.enableRipple(true);
var items = [
    {
        text: "Home",
        url: "../"
    },
    {
        text: "Getting",
        url: "../breadcrumb/getting-started"
    },
    {
        text: "Data-Binding",
        url: "../breadcrumb/data-binding"
    },
    {
        text: "Icons",
        url: "../breadcrumb/icons"
    },
    {
        text: "Navigation",
        url: "../breadcrumb/navigation"
    },
]
```

```

        {
            text: "templates",
            url: "../breadcrumb/templates"
        },
        {
            text: "Overflow",
            url: "../breadcrumb/overflow"
        }
    ];
    new ej.navigations.Breadcrumb({
        items: items,
        enableNavigation: false,
        maxItems: 3,
        overflowMode: 'Hidden'
    }, '#breadcrumb');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Essential JS 2">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <!--style reference from app-->
    <link href="styles.css" rel="stylesheet">
    <!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="control-section">
            <nav id="breadcrumb"></nav>
        </div>
    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

None

None mode shows all the items in a single line.

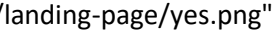
Accessibility in `##Platform_Name##` Breadcrumb control

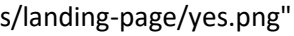
The Breadcrumb component followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

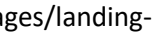
The accessibility compliance for the Breadcrumb component is outlined below.

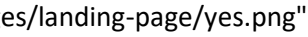
| Accessibility Criteria | Compatibility |

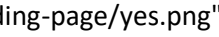
| -- | -- |

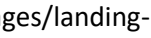
| [WCAG 2.2](#) Support |  |


| [Section 508](#) Support |  |

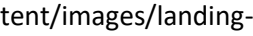
| Screen Reader Support |  |

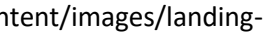
| Right-To-Left Support |  |

| Color Contrast |  |

| Mobile Device Support |  |

| Keyboard Navigation Support |  |

| [Accessibility Checker](#) Validation |  |

| [Axe-core](#) Accessibility Validation |  |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<div> - All features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>

WAI-ARIA attributes

The Breadcrumb component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Breadcrumb component:

| Attributes | Purpose |

| --- | --- |

| `aria-label` | Indicates the breadcrumb item text. |

| `aria-disabled` | Indicates the state of breadcrumb item whether it is disabled. |

Keyboard interaction

The Breadcrumb component followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Breadcrumb component.

| Press | To do this |

| --- | --- |

| Tab | Navigate to the next item and also next item in the popup of menu type overflow. |

| Shift + Tab | Navigate to the previous item also previous item in the popup of menu type overflow. |

| Enter key in normal mode | Select the breadcrumb item. |

| Enter key in normal mode | To open the popup of menu type overflow mode when you press enter on collapsed button and It will expand the items of collapsed type overflow mode when you press enter on collapsed button. |

Ensuring accessibility

The Breadcrumb component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Breadcrumb component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Breadcrumb component with accessibility tools.

See also

- [Accessibility in Syncfusion ##Platform_Name## components](#)

Bullet Chart

Bullet chart dimensions in ##Platform_Name## Bullet chart control

Size for Container

The size of the Bullet Chart is determined by the container size, and it can be changed inline or via CSS as following.

```
<div id='container'>
<div id='element' style="width:650px; height:350px;"></div>
</div>
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Size for Bullet Chart

The **width** and **height** properties are used to adjust the size of the Bullet Chart.

Pixel

Can set the size of the Bullet Chart in pixels as shown below.

INDEX.JS

```

var chart = new ej.charts.BulletChart({
  width: '650', height: '350',
  dataSource: [{ value: 23, target: 22 }],
  animation: { enable: false },
  valueField: 'value',
  targetField: 'target',
  ranges: [{ end: 20 },
    { end: 25 },
    { end: 30 }
  ],
  minimum: 0, maximum: 30, interval: 5,
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>

```



```

<title>EJ2 Animation</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Percentage

By setting a value in percentage, the Bullet Chart gets its dimension with respect to its container. For example, when the height is **50%**, the Bullet Chart renders to half of the container's height.

INDEX.JS

```

var chart = new ej.charts.BulletChart({
    width: '80%', height: '90%',
    dataSource: [{ value: 23, target: 22 }],
    animation: { enable: false },
    valueField: 'value',
    targetField: 'target',
    ranges: [{ end: 20 },
        { end: 25 },
        { end: 30 }
    ],
    minimum: 0, maximum: 30, interval: 5,
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
<title>EJ2 Animation</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

If the size is not specified, the Bullet Chart will be rendered with a height of **126px** and a width of the window.

Axis customization in ##Platform_Name## Bullet chart control

MajorTickLines and MinorTickLines Customization

You can customize the **width**, **color**, and **size** of minor and major tick lines using the [majorTickLines](#) and [minorTickLines](#) properties of the bullet-chart.

The following properties can be used to customize **majorTicklines** and **minorTicklines**.

- **width** - Specifies the width of ticklines.
- **height** - Specifies the height of ticklines.
- **color** - Specifies the color of ticklines.
- **useRangeColor** - Specifies the color of ticklines and represents the color from corresponding range colors.

INDEX.JS

```

var chart = new ej.charts.BulletChart({
    majorTickLines: { color: 'blue', width: 5 },
    minorTickLines: { width: 4, color: 'red' },
    dataSource: [{ value: 23, target: 22 }],
    animation: { enable: false },
    valueField: 'value',
    targetField: 'target',
    ranges: [{ end: 20 },
        { end: 25 },
        { end: 30 }
    ],
    minimum: 0, maximum: 30, interval: 5
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Tick Placement

You can place major and minor ticks **inside** or **outside** the ranges using the [tickPosition](#) property of bullet-chart. The major and the minor ticks can be placed **inside** or **outside** the ranges using the [tickPosition](#) property.

INDEX.JS

```

var chart = new ej.charts.BulletChart({
  tickPosition: 'Inside',
  dataSource: [{ value: 23, target: 22 }],
  animation: { enable: false },
  valueField: 'value',
  targetField: 'target',
  ranges: [{ end: 20 },
    { end: 25 },
    { end: 30 }
  ],
  minimum: 0, maximum: 30, interval: 5,
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">

```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Label Format

Axis Label Format

Axis numeric labels can be formatted by using the [labelFormat](#) property. Axis labels support all globalize formats. The following table describes the result of applying some commonly used label formats on numeric axis values.

Axis numeric labels can be formatted by using the `labelFormat` property. Axis labels support all globalize formats.

INDEX.JS

```

var chart = new ej.charts.BulletChart({
    labelFormat: 'c',
    dataSource: [{ value: 23, target: 22 }],
    animation: { enable: false },
    valueField: 'value',
    targetField: 'target',
    ranges: [{ end: 20 },
        { end: 25 },
        { end: 30 }
    ],
    minimum: 0, maximum: 30, interval: 5,
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
<title>EJ2 Animation</title>
<meta charset="utf-8">

```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The following table describes the result of applying some commonly used formats to numeric axis labels.

<!-- markdownlint-disable MD033 -->

Label Value	Label Format property value	Result	Description
1000	n1	1000.0	The Number is rounded to 1 decimal place
1000	n2	1000.00	The Number is rounded to 2 decimal place
1000	n3	1000.000	The Number is rounded to 3 decimal place
0.01	p1	1.0%	The Number is converted to percentage with 1 decimal place
0.01	p2	1.00%	The Number is converted to percentage with 2 decimal place
0.01	p3	1.000%	The Number is converted to percentage with 3 decimal place
1000	c1	\$1000.0	The Currency symbol is appended to number and number is rounded to 1 decimal place
1000	c2	\$1000.00	The Currency symbol is appended to number and number is rounded to 2 decimal place

GroupingSeparator

To separate groups of thousands, use the [enableGroupSeparator](#) property of bullet-chart.

To separate the groups of thousands, set the `enableGroupSeparator` property to `true`.

INDEX.JS

```
var chart = new ej.charts.BulletChart({
  enableGroupSeparator: true,
  title: 'Sales Rate',
  dataSource: [{ value: 1500, target: 1300 }],
  animation: { enable: false },
  valueField: 'value',
  targetField: 'target',
  ranges: [{ end: 500 },
    { end: 1500 },
    { end: 2500 }
  ],
  minimum: 0, maximum: 2500, interval: 250
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Custom Label Format

Using the `labelFormat` property, axis labels can be specified with a custom defined format in addition to the axis value. The label format uses a placeholder such as `${value}K`, which represents the axis label.

INDEX.JS

```
var chart = new ej.charts.BulletChart({
  labelFormat: '${value}K',
```

```

    title: 'Sales Rate',
    dataSource: [{ value: 1500, target: 1300, category: 'Product A' }],
    animation: { enable: false },
    valueField: 'value',
    targetField: 'target',
    ranges: [{ end: 500 },
              { end: 1500 },
              { end: 2500 }
    ],
    minimum: 0, maximum: 2500, interval: 250
  }, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Label Placement

You can customize the axis labels **inside** or **outside** the bullet-chart using the [labelPosition](#) property. Label can be placed **Inside** or **Outside** of the ranges using the [labelPosition](#) property.

INDEX.JS

```

var chart = new ej.charts.BulletChart({
  labelPosition: 'Inside',
  dataSource: [{ value: 23, target: 22 }],
  animation: { enable: false },
  valueField: 'value',
  targetField: 'target',
  ranges: [{ end: 20 },

```

```

        { end: 25 },
        { end: 30 }
    ],
    minimum: 0, maximum: 30, interval: 5,
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Opposed Position

To place an axis opposite to its original position, set the [opposedPosition](#) property to true. To place an axis opposite to its original position, set the `opposedPosition` property to **true**.

INDEX.JS

```

var chart = new ej.charts.BulletChart({
  majorTickLines: { color: 'blue', width: 5 },
  minorTickLines: { width: 4, color: 'red' },
  dataSource: [{ value: 23, target: 22 }],
  animation: { enable: false },
  valueField: 'value',
  targetField: 'target',
  ranges: [{ end: 20 },
    { end: 25 },
    { end: 30 }
  ],
  minimum: 0, maximum: 30, interval: 5
}, '#element');

```


INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Category Label

The Bullet Chart supports X-axis label by specifying the property from the data source to the categoryField. It helps to understand the input data in a more efficient way.

INDEX.JS

```

var chart = new ej.charts.BulletChart({
  title: 'Sales Rate',
  dataSource: [{ value: 1500, target: 1300, category: 'Product A' }],
  animation: { enable: false },
  valueField: 'value',
  targetField: 'target',
  categoryField: 'category',
  ranges: [{ end: 500 },
    { end: 1500 },
    { end: 2500 }
  ],
  minimum: 0, maximum: 2500, interval: 250
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>

```

```

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Category Label Customization

The label color, opacity, font size, font family, font weight, and font style can be customized by using the `categoryLabelStyle` setting for category and the `labelStyle` setting for axis label. The `useRangeColor` property specifies the color of the axis label and represents the color from the corresponding range colors.

INDEX.JS

```

var chart = new ej.charts.BulletChart({
    title: 'Sales Rate',
    dataSource: [{ value: 1500, target: 1300, category: 'Product A' }],
    animation: { enable: false },
    valueField: 'value',
    targetField: 'target',
    categoryField: 'category',
    categoryLabelStyle: { color: 'Orange' },
    ranges: [{ end: 500 },
        { end: 1500 },
        { end: 2500 }
    ],
    minimum: 0, maximum: 2500, interval: 250
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
<title>EJ2 Animation</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">

```

```

<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

<!-- markdownlint-disable MD036 -->

Data binding in ##Platform_Name## Bullet chart control

The `dataSource` property accepts a collection of values as input that helps to display measures, and compares them to a target bar. To display the actual and target bar, specify the property from the datasource into the `valueField` and `targetField` respectively.

INDEX.JS

```

var localData = [
    {
        value: 5, comparativeMeasureValue: 7.5,
        category: '2001'
    },
    {
        value: 7, comparativeMeasureValue: 5,
        category: '2002'
    },
    {
        value: 10, comparativeMeasureValue: 6,
        category: '2003'
    },
    {
        value: 5, comparativeMeasureValue: 8,
        category: '2004'
    },
    {
        value: 12, comparativeMeasureValue: 5,
        category: '2005'
    },
    {
        value: 8, comparativeMeasureValue: 6,

```

```

        category: '2006'
    }
    ];
    var chart = new ej.charts.BulletChart({
        dataSource: localData,
        animation: { enable: false },
        valueField: 'value',
        targetField: 'comparativeMeasureValue',
        title: 'Profit in %',
        height: '400',
        ranges: [{ end: 5 },
            { end: 15 },
            { end: 20 }
        ],
        minimum: 0, maximum: 20, interval: 5,
    }, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
    <script>
    var ele = document.getElementById('container');
    if(ele) {
        ele.style.visibility = "visible";
    }
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

<!-- markdownlint-disable MD036 -->

Ranges in ##Platform_Name## Bullet chart control

Ranges represent the quality of a specific range such as **Good**, **Bad** and **Satisfactory** in the Bullet Chart scale. The ending point of a qualitative range is specified in the **end** property in **ranges**. The **minimum** value of a quantitative scale is considered the starting point of the first range or the previous range end point.

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Color Customization

Enhance the readability of ranges with color and opacity. It can be applied using the **color** and **opacity** properties in **ranges**.

INDEX.JS

```

var chart = new ej.charts.BulletChart({
  title: 'Sales Rate',
  dataSource: [
    { value: 55, target: 75, category: 'Year 1' },
    { value: 70, target: 70, category: 'Year 2' },
    { value: 85, target: 75, category: 'Year 3' }
  ],
  animation: { enable: false },
  valueField: 'value',
  targetField: 'target',
  categoryField: 'category',
  categoryLabelStyle: { color: 'red', size: '13', fontWeight: 'bold' },
  ranges: [ { end: 35, color: 'darkred', opacity: 0.5 },
    { end: 50, color: 'red', opacity: 1 },
    { end: 75, color: 'blue', opacity: 0.7 },
    { end: 90, color: 'lightgreen', opacity: 1 },
    { end: 100, color: 'green', opacity: 1 }
  ],
  minimum: 0, maximum: 100, interval: 10,
  height: '400'
});

```

```
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

<!-- markdownlint-disable MD036 -->

Value bar in ##Platform_Name## Bullet chart control

To display the primary data or the current value of the data being measured known as the **Feature Measure** that should be encoded as a bar. This is called as the **Actual Bar** or the **Feature Bar** in the Bullet Chart, and to display the actual bar the [valueField](#) should be mapped to the appropriate field from the data source.

INDEX.JS

```
var chart = new ej.charts.BulletChart({
  title: 'Sales Rate',
  dataSource: [
    { value: 55, target: 75, category: 'Year 1' },
  ],
  animation: { enable: false },
  valueField: 'value',
  ranges: [ { end: 35 },
    { end: 50 },
    { end: 100 }
  ],
  minimum: 0, maximum: 100, interval: 20
}, '#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Types of actual bar

The shape of the actual bar can be customized using the [type](#) property of the Bullet Chart. The actual bar contains **Rect** and **Dot** shapes. By default, the actual bar shape is Rect.

INDEX.JS

```

var chart = new ej.charts.BulletChart({
  title: 'Sales Rate',
  dataSource: [
    { value: 55, target: 75, category: 'Year 1' },
  ],
  animation: { enable: false },
  valueField: 'value',
  ranges: [ { end: 35 },
    { end: 50 },
    { end: 100 }
  ],
  type: 'Dot',
  minimum: 0, maximum: 100, interval: 20
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>

```

```

<title>EJ2 Animation</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Actual bar customization

Border customization

Using the [valueBorder](#) property of the bullet chart, you can customize the border [color](#) and [width](#) of the actual bar.

INDEX.JS

```

var chart = new ej.charts.BulletChart({
    title: 'Sales Rate',
    dataSource: [
        { value: 55, target: 75, category: 'Year 1' },
    ],
    animation: { enable: false },
    valueField: 'value',
    ranges: [ { end: 35 },
        { end: 50 },
        { end: 100 }
    ],
    valueBorder: { color: 'red', width: 3 },
    minimum: 0, maximum: 100, interval: 20
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
<title>EJ2 Animation</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">

```



```

<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Fill color and height customization

Customize the fill color and height of the actual bar using the [valueFill](#) and [valueHeight](#) properties of the bullet chart. Also, you can bind the color for the actual bar from [dataSource](#) for the bullet chart using [valueFill](#) property.

INDEX.JS

```

var chart = new ej.charts.BulletChart({
    title: 'Sales Rate',
    dataSource: [
        { value: 55, target: 75, category: 'Year 1', color: 'blue'
    },
    ],
    animation: { enable: false },
    valueField: 'value',
    ranges: [
        { end: 35 },
        { end: 50 },
        { end: 100 }
    ],
    valueFill: 'color',
    valueHeight: 15,
    minimum: 0, maximum: 100, interval: 20
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

```

```

<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

<!-- markdownlint-disable MD036 -->

Comparative bar in ##Platform_Name## Bullet chart control

The line marker that runs perpendicular to the orientation of the graph is known as the **Comparative Measure** and it is used as a target marker to compare against the feature measure value. This is also called as the **Target Bar** in the Bullet Chart. To display the target bar, the [targetField](#) should be mapped to the appropriate field from the datasource.

INDEX.JS

```

var chart = new ej.charts.BulletChart({
    title: 'Sales Rate',
    dataSource: [
        { value: 55, target: 75, category: 'Year 1' },
    ],
    animation: { enable: false },
    targetField: 'target',
    ranges: [ { end: 35 },
        { end: 50 },
        { end: 100 }
    ],
    minimum: 0, maximum: 100, interval: 20
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
<title>EJ2 Animation</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">

```

```

<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Types of target bar

The shape of the target bar can be customized using the [targetTypes](#) property and it supports **Circle**, **Cross**, and **Rect** shapes. The default type of the target bar is **Rect**.

INDEX.JS

```

var chart = new ej.charts.BulletChart({
    title: 'Sales Rate',
    dataSource: [
        { value: 55, target: 75, category: 'Year 1' },
    ],
    animation: { enable: false },
    targetField: 'target',
    targetTypes: ['Circle'],
    ranges: [ { end: 35 },
        { end: 50 },
        { end: 100 }
    ],
    minimum: 0, maximum: 100, interval: 20
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Target bar customization

The following properties can be used to customize the target bar. Also, you can bind the color for the target bar from [dataSource](#) for the bullet chart.

- [targetColor](#) - Specifies the fill color of target bar.
- [targetWidth](#) - Specifies the width of target bar.

INDEX.JS

```

var chart = new ej.charts.BulletChart({
    title: 'Sales Rate',
    dataSource: [
        { value: 55, target: 75, category: 'Year 1', color: 'red' },
    ],
    animation: { enable: false },
    targetField: 'target',
    targetColor: 'color',
    targetWidth: 15,
    ranges: [
        { end: 35 },
        { end: 50 },
        { end: 100 }
    ],
    minimum: 0, maximum: 100, interval: 20
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

<!-- markdownlint-disable MD036 -->

Title in ##Platform_Name## Bullet chart control

Title

The title of the Bullet Chart displays the information about the data plotted by specifying it in the `title` property.

INDEX.JS

```

var chart = new ej.charts.BulletChart({
    title: 'Sales Rate',
    dataSource: [
        { value: 55, target: 75, category: 'Year 1' },
    ],
    animation: { enable: false },
    targetField: 'target',
    ranges: [ { end: 35 },
        { end: 50 },
        { end: 100 }
    ],
    minimum: 0, maximum: 100, interval: 20
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Subtitle

To show additional information about the data plotted, the Bullet Chart can also be given a subtitle using the `subtitle` property.

INDEX.JS

```

var chart = new ej.charts.BulletChart({
    title: 'Sales Rate in dollars',
    subtitle: '(in dollars $)',
    dataSource: [
        { value: 55, target: 45, category: 'Year 1' },
    ],
    animation: { enable: false },
    targetField: 'target',
    valueField: 'value',
    labelFormat: '${value}',
    ranges: [ { end: 35 },
        { end: 50 },
        { end: 100 }
    ],
    minimum: 0, maximum: 100, interval: 20
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>

```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Title and SubTitle Position

The title and the subtitle positions can be customized using the `titlePosition` property. Possible positions are **Left**, **Right**, **Top**, and **Bottom**.

Position as Left

By setting the `titlePosition` to **Left**, you can display the title and subtitle at the left side of the Bullet Chart.

INDEX.JS

```

var chart = new ej.charts.BulletChart({
    title: 'Sales Rate',
    subtitle: '(in dollars $)',
    dataSource: [
        { value: 55, target: 45, category: 'Year 1' },
    ],
    animation: { enable: false },
    targetField: 'target',
    valueField: 'value',
    titlePosition: 'Left',
    labelFormat: '${value}',
    ranges: [ { end: 35 },
        { end: 50 },
        { end: 100 }
    ],
    minimum: 0, maximum: 100, interval: 20
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Position as Right

By setting the **titlePosition** to **Right**, you can display the title and subtitle at the right side of the Bullet Chart.

INDEX.JS

```

var chart = new ej.charts.BulletChart({
    title: 'Sales Rate',
    subtitle: '(in dollars $)',
    dataSource: [
        { value: 55, target: 45, category: 'Year 1' },
    ],
    animation: { enable: false },
    targetField: 'target',
    valueField: 'value',
    titlePosition: 'Right',
    labelFormat: '${value}',
    ranges: [{ end: 35 },
        { end: 50 },
        { end: 100 }
    ],
    minimum: 0, maximum: 100, interval: 20
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

```



```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Position as Top

By setting the **titlePosition** to **Top**, you can display the title and subtitle at the top of the Bullet Chart. The default title and subtitle positions of the Bullet Chart is **Top**.

INDEX.JS

```

var chart = new ej.charts.BulletChart({
    title: 'Sales Rate',
    subtitle: '(in dollars $)',
    dataSource: [
        { value: 55, target: 45, category: 'Year 1' },
    ],
    animation: { enable: false },
    targetField: 'target',
    valueField: 'value',
    titlePosition: 'Top',
    labelFormat: '${value}',
    ranges: [ { end: 35 },
        { end: 50 },
        { end: 100 }
    ],
    minimum: 0, maximum: 100, interval: 20
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Position as Bottom

By setting the **titlePosition** to **Bottom**, you can display the title and subtitle at the bottom of the Bullet Chart.

INDEX.JS

```

var chart = new ej.charts.BulletChart({
    title: 'Sales Rate',
    subtitle: '(in dollars $)',
    dataSource: [
        { value: 55, target: 45, category: 'Year 1' },
    ],
    animation: { enable: false },
    targetField: 'target',
    valueField: 'value',
    titlePosition: 'Bottom',
    labelFormat: '${value}',
    ranges: [ { end: 35 },
        { end: 50 },
        { end: 100 }
    ],
    minimum: 0, maximum: 100, interval: 20
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Title Customization

The title color, opacity, font size, font family, font weight, and font style can be customized using the `titleStyle` property.

INDEX.JS

```

var chart = new ej.charts.BulletChart({
    title: 'Sales Rate',
    dataSource: [
        { value: 55, target: 45, category: 'Year 1' },
    ],
    animation: { enable: false },
    targetField: 'target',
    valueField: 'value',
    labelFormat: '${value}',
    titleStyle: { size: '22', color: 'red', fontFamily: 'Italic',
fontWeight: 'Bold'},
    ranges: [ { end: 35 },
        { end: 50 },
        { end: 100 }
    ],
    minimum: 0, maximum: 100, interval: 20
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

SubTitle Customization

The sub-title color, opacity, font size, font family, font weight, and font style can be customized using the `subtitleStyle` property.

INDEX.JS

```

var chart = new ej.charts.BulletChart({
    title: 'Sales Rate',
    dataSource: [
        { value: 55, target: 45, category: 'Year 1' },
    ],
    animation: { enable: false },
    targetField: 'target',
    valueField: 'value',
    labelFormat: '${value}',
    subtitleStyle: { size: '22', color: 'red', fontFamily: 'Italic',
fontWeight: 'Bold'},
    ranges: [ { end: 35 },
        { end: 50 },
        { end: 100 }
    ],
    minimum: 0, maximum: 100, interval: 20
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

<!-- markdownlint-disable MD036 -->

Customization in ##Platform_Name## Bullet chart control

Orientation

The Bullet Chart can be rendered in different orientations such as **Horizontal** or **Vertical** via the **orientation** property. By default, the Bullet Chart is rendered in the **Horizontal** orientation.

INDEX.JS

```

var chart = new ej.charts.BulletChart({
    title: 'Sales Rate in dollars',
    subtitle: '(in dollars $)',
    dataSource: [
        { value: 55, target: 45, category: 'Year 1' },
    ],
    animation: { enable: false },
    targetField: 'target',
    valueField: 'value',
    labelFormat: '${value}',
    ranges: [ { end: 35 },
        { end: 50 },
        { end: 100 }
    ],
    width: '20%',
    orientation: 'Vertical',
    minimum: 0, maximum: 100, interval: 20
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">

```

```

<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Right-to-left (RTL)

The Bullet Chart supports the right-to-left rendering that can be enabled by setting the `enableRtl` property to **true**.

INDEX.JS

```

var chart = new ej.charts.BulletChart({
    dataSource: [
        { value: 1500, target: 1000, category: 'Year 1' },
    ],
    animation: { enable: false },
    targetField: 'target',
    valueField: 'value',
    ranges: [{ end: 500 },
        { end: 1500 },
        { end: 2000 }
    ],
    enableRtl: true,
    minimum: 0, maximum: 2000, interval: 200
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Animation

The actual and the target bar supports the linear animation via the **animation** setting. The speed and the delay are controlled using the **duration** and **delay** properties respectively.

INDEX.JS

```

var chart = new ej.charts.BulletChart({
    dataSource: [
        { value: 1500, target: 1000, category: 'Year 1' },
    ],
    animation: { enable: true },
    targetField: 'target',
    valueField: 'value',
    ranges: [ { end: 500 },
        { end: 1500 },
        { end: 2000 }
    ],
    minimum: 0, maximum: 2000, interval: 200
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>

```

```

<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Theme

The Bullet Chart supports different type of themes via the `theme` property.

INDEX.JS

```

var chart = new ej.charts.BulletChart({
    title: 'Profit in %',
    dataSource: [
        { value: 50, target: 45, category: 'Year 1' },
    ],
    animation: { enable: false },
    targetField: 'target',
    valueField: 'value',
    ranges: [ { end: 15 },
        { end: 50 },
        { end: 100 }
    ],
    theme: 'HighContrast',
    minimum: 0, maximum: 100, interval: 10
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>

```



```

</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

<!-- markdownlint-disable MD036 -->

Data label in ##Platform_Name## Bullet chart control

Data Labels are used to identify the value of actual bar in the Bullet Chart component. The Data Labels will be shown by specifying the `dataLabel` setting's `enable` property to **true**.

INDEX.JS

```

var localData = [
    {
        value: 5, comparativeMeasureValue: 7.5,
        category: '2001'
    },
    {
        value: 7, comparativeMeasureValue: 5,
        category: '2002'
    },
    {
        value: 10, comparativeMeasureValue: 6,
        category: '2003'
    },
    {
        value: 5, comparativeMeasureValue: 8,
        category: '2004'
    },
    {
        value: 12, comparativeMeasureValue: 5,
        category: '2005'
    },
    {
        value: 8, comparativeMeasureValue: 6,
        category: '2006'
    }
];
var chart = new ej.charts.BulletChart({
    dataSource: localData,
    animation: { enable: false },
    valueField: 'value',
    targetField: 'comparativeMeasureValue',
    title: 'Profit in percentage',
    height: '400',
    ranges: [{ end: 5 },
        { end: 15 },
        { end: 20 }
    ],
    labelFormat: '{value}%',

```

```
dataLabel: { enable: true },
minimum: 0, maximum: 20, interval: 5,
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Data Label Customization

Data Labels color, opacity, font size, font family, font weight, and font style can be customized using the `labelStyle`.

INDEX.JS

```
var localData = [
  {
    value: 5, comparativeMeasureValue: 7.5,
    category: '2001'
  },
  {
    value: 7, comparativeMeasureValue: 5,
    category: '2002'
  },
  {
    value: 10, comparativeMeasureValue: 6,
    category: '2003'
  },
  {
    value: 5, comparativeMeasureValue: 8,
```

```

        category: '2004'
    },
    {
        value: 12, comparativeMeasureValue: 5,
        category: '2005'
    },
    {
        value: 8, comparativeMeasureValue: 6,
        category: '2006'
    }
];
var chart = new ej.charts.BulletChart({
    dataSource: localData,
    animation: { enable: false },
    valueField: 'value',
    targetField: 'comparativeMeasureValue',
    title: 'Profit in percentage',
    height: '400',
    ranges: [{ end: 5 },
        { end: 15 },
        { end: 20 }
    ],
    labelFormat: '{value}%',
    dataLabel: { enable: true, labelStyle: { color: 'yellow', size: '20' }
},
    minimum: 0, maximum: 20, interval: 5,
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
<script src="index.js" type="text/javascript"></script>

```

```
</body></html>
```

Tool tip in ##Platform_Name## Bullet chart control

When the mouse is hovered over a bar in the Bullet Chart, the tooltip displays important summary about the actual and the target bar values.

Default tooltip

By setting [enable](#) the property to 'True' and by injecting `BulletTooltip` module using `BulletChart.Inject(BulletTooltip)`. The 'Tooltip' is visible in the 'Bullet chart' by default.

The tooltip is not visible by default. To make it visible, set the `enable` property in the `tooltip` to `true` and injecting `BulletTooltip` module using `BulletChart.Inject(BulletTooltip)`.

INDEX.JS

```
var localData = [
  {
    value: 5, comparativeMeasureValue: 7.5,
    category: '2001'
  },
  {
    value: 7, comparativeMeasureValue: 5,
    category: '2002'
  },
  {
    value: 10, comparativeMeasureValue: 6,
    category: '2003'
  },
  {
    value: 5, comparativeMeasureValue: 8,
    category: '2004'
  },
  {
    value: 12, comparativeMeasureValue: 5,
    category: '2005'
  },
  {
    value: 8, comparativeMeasureValue: 6,
    category: '2006'
  }
];
var chart = new ej.charts.BulletChart({
  dataSource: localData,
  animation: { enable: false },
  valueField: 'value',
  targetField: 'comparativeMeasureValue',
  title: 'Profit in %',
  height: '400',
  ranges: [{ end: 5 },
    { end: 15 },
    { end: 20 }
  ],
  minimum: 0, maximum: 20, interval: 5,
}, '#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script id="Tooltip" type="text/x-template">
    <div id="wrap">
      <table style="width:100%; background-color: #ffffff; border-
      spacing: 0px; border-collapse:separate; border: 1px solid grey; border-
      radius:10px; padding-top: 5px; padding-bottom:5px">
        <tr>
          <td style="font-weight:bold; color:black; padding-left:
          5px;padding-top: 2px;padding-bottom: 2px;">Sales</td>
        </tr>
        <tr>
          <td style="padding-left: 5px; color:black; padding-
          right: 5px; padding-bottom: 2px;">Target    : ${target}K </td>
        </tr>
        <tr>
          <td style="padding-left: 5px; color:black; padding-
          right: 5px">Current : ${value}K </td>
        </tr>
      </table>
    </div>
  </script>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Tooltip template

Any HTML elements can be displayed in the tooltip by using the **template** property of the **tooltip**. You can use the **\${target}** and **\${value}** as place holders in the HTML element to display the value and target values from the data source of corresponding data point.

INDEX.JS

```

var chart = new ej.charts.BulletChart({
  height: '110px',
  tooltip: { enable: true, template : '#Tooltip' },
  dataSource: [{ value: 70, target: 50 }],
  valueField: 'value',
  targetField: 'target',
  animation: { enable: false },
  ranges: [{ end: 30, color: '#599C20' },
    { end: 60, color: '#EFC820' },
    { end: 100, color: '#CA4218' }
  ],
  minimum: 0, maximum: 100, interval: 10,
  title: 'Revenue YTD',
  labelFormat: '${value}K',
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script id="Tooltip" type="text/x-template">
    <div id="wrap">
      <table style="width:100%; background-color: #ffffff; border-
spacing: 0px; border-collapse:separate; border: 1px solid grey; border-
radius:10px; padding-top: 5px; padding-bottom:5px">
        <tr>
          <td style="font-weight:bold; color:black; padding-left:
5px;padding-top: 2px;padding-bottom: 2px;">Sales</td>
        </tr>
        <tr>
          <td style="padding-left: 5px; color:black; padding-
right: 5px; padding-bottom: 2px;">Target    : ${target}K </td>
        </tr>
        <tr>
          <td style="padding-left: 5px; color:black; padding-
right: 5px">Current : ${value}K </td>
        </tr>

```

```

        </table>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customization of the appearance of tooltip

The [fill](#) and [border](#) properties are used to customize the background color and border of the tooltip respectively. The [textStyle](#) property in the tooltip is used to customize the font of the tooltip text.

The following properties can be used to customize the Bullet Chart tooltip.

- **fill** - Specifies the color of tooltip.
- **border** - Specifies the tooltip border color and width.
- **textStyle** - Specifies the tooltip font family, font style, font weight, color and size.

INDEX.JS

```

var piechart = new ej.charts.AccumulationChart({
    series: [
        {
            dataSource: [
                { x: 'Chrome', y: 61.3, text: 'Chrome: 61.3%' },
                { x: 'Safari', y: 24.6, text: 'Safari: 24.6%' },
                { x: 'Edge', y: 5.0, text: 'Edge: 5.0%' },
                { x: 'Samsung Internet', y: 2.7, text: 'Samsung Internet: 2.7%' },
                { x: 'Firefox', y: 2.6, text: 'Firefox: 2.6%' },
                { x: 'Others', y: 3.6, text: 'Others: 3.6%' }
            ],
            innerRadius: '65%',
            xName: 'x',
            yName: 'y'
        },
        {
            centerLabel: {
                text: 'Mobile<br>Browsers<br>Statistics',
                textStyle: {
                    fontWeight: 600
                }
            }
        }
    ],
    '#element'
});

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

```

```

<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
    <script id="Unemployment" type="text/x-template">
        <div id='templateWrap'>
            <table style="width:100%; border: 1px solid black;">
            <tr><th colspan="2" bgcolor="#00FFFF">Unemployment</th></tr>
            <tr><td bgcolor="#00FFFF">${x}:</td><td
bgcolor="#00FFFF">${y}</td></tr>
            </table>
        </div>
    </script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Accessibility in ##Platform_Name## Bullet chart control

The Bullet chart control followed the accessibility guidelines and standards, including [ADA](#), [Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Bullet chart control is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support | |

| [Section 508](#) Support | |

| Screen Reader Support | |

| Right-To-Left Support | |

| Color Contrast | |

| Mobile Device Support | |

| Keyboard Navigation Support | |

| [Accessibility Checker](#) Validation | |

| [Axe-core](#) Accessibility Validation | |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<div> - All features of the control meet the requirement.</div>

<div> - Some features of the control do not meet the requirement.</div>

<div> - The control does not meet the requirement.</div>

WAI-ARIA attributes

The Bullet chart control followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Bullet chart control:

- img (role)
- button (role)
- aria-label (attribute)
- aria-pressed (attribute)

Keyboard interaction

The Bullet chart control followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Bullet chart control.

| **Press** | **To do this** |

| --- | --- |

| **Tab** | Moves the focus to the next element in the Bullet chart. |

| Shift + Tab | Moves the focus to the previous element in the Bullet chart. |

| Ctrl + P | Prints the Bullet chart. |

Ensuring accessibility

The Bullet chart control's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Bullet chart control is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Bullet chart control with accessibility tools.

See also

- [Accessibility in Syncfusion ##Platform_Name## controls](#)

ButtonGroup

Getting started in ##Platform_Name## Button group control

This section explains how to create a simple ButtonGroup, and configure its available functionalities by using the Essential JS 2 [quickstart](#) seed repository.

This application is integrated with the `webpack.config.js` configuration and uses the latest version of the [webpack-cli](#). It requires node `v14.15.0` or higher. For more information about webpack and its features, refer to the [webpack documentation](#).

Dependencies

The following list of dependencies are required to use the ButtonGroup component in your application.

`js

|-- @syncfusion/ej2-splitbuttons

,

Set up development environment

Open the command prompt from the required directory, and run the following command to clone the Syncfusion JavaScript (Essential JS 2) quickstart project from [GitHub](#).

CMD

```
git clone https://github.com/SyncfusionExamples/ej2-quickstart-webpack- ej2-quickstart
```

After cloning the application in the `ej2-quickstart` folder, run the following command line to navigate to the `ej2-quickstart` folder.

CMD

```
cd ej2-quickstart
```

Add Syncfusion JavaScript packages

Syncfusion JavaScript (Essential JS 2) packages are available on the [npmjs.com](#) public registry. You can install all Syncfusion JavaScript (Essential JS 2) controls in a single [@syncfusion/ej2](#) package or individual packages for each control.

The quickstart application is preconfigured with the dependent [@syncfusion/ej2](#) package in the `~/package.json` file. Use the following command to install the dependent npm packages from the command prompt.

NPM

```
npm install
```

Import the Syncfusion CSS styles

The ButtonGroup CSS files are available in the ej2-splitbuttons package folder. This can be referenced in the `~/src/styles/styles.css` file of your application by using the following code.

STYLE.CSS

```
@import '../..../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../..../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../..../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
```

Add ButtonGroup to the project

Add the HTML div tag with class name as `e-btn-group` and the button elements that should group inside the div element with class name as

`e-btn` to your `index.html` file.

[src/index.html]

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>Essential JS 2</title>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-scale=1.0, user-scalable=no" />
<meta name="description" content="Essential JS 2" />
<meta name="author" content="Syncfusion" />
<link rel="shortcut icon" href="resources/favicon.ico" />
<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet" />
</head>
<body>
<div>
<!--element which is going to render-->
<div class='e-btn-group'>
<button class='e-btn'>HTML</button>
<button class='e-btn'>CSS</button>
<button class='e-btn'>Javascript</button>
</div>
</div>
</body>
</html>
```

Run the application

Run the application in the browser by using the following command.

NPM

```
npm start
```

The following example shows a basic ButtonGroup component.

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Button-Group</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="e-btn-group">
      <button class="e-btn">HTML</button>
      <button class="e-btn">CSS</button>
      <button class="e-btn">Javascript</button>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

STYLES.CSS

```
#container {
  visibility: hidden;
}
```

```
#loader {
  color: #008cff;
  height: 40px;
  left: 45%;
  position: absolute;
  top: 45%;
  width: 30%;
}
.e-btn-group {
  margin: 25px 5px 20px 20px;
}
```

Orientation

ButtonGroup can be arranged in a vertical and horizontal orientation. By default, it is horizontally aligned.

Vertical Orientation

ButtonGroup can be aligned vertically by using the built-in CSS class `e-vertical` to the target element.

The following example illustrates how to achieve vertical orientation in ButtonGroup.

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Button-Group</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="e-btn-group e-vertical">
      <button class="e-btn">HTML</button>
      <button class="e-btn">CSS</button>
      <button class="e-btn">Javascript</button>
    </div>
  </div>
<script>
var ele = document.getElementById('container');
```

```

if(ele) {
    ele.style.visibility = "visible";
}

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.e-btn-group {
    margin: 25px 5px 20px 20px;
}

```

ButtonGroup does not support SplitButton component nesting in a vertical orientation.

See Also

- [Initialize ButtonGroup using util function](#)

Types and styles in ##Platform_Name## Button group control

This section explains about different types and styles of ButtonGroup.

ButtonGroup types

Outline ButtonGroup

An Outline ButtonGroup has a border with transparent background. To create an Outline ButtonGroup, `e-outline` class should be added to the target element and also add `e-outline` and `e-btn` class to each button elements.

The following sample illustrates how to achieve an Outline ButtonGroup.

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Button-Group</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="e-btn-group e-outline">
            <button class="e-btn e-outline">HTML</button>
            <button class="e-btn e-outline">CSS</button>
            <button class="e-btn e-outline">Javascript</button>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.e-btn-group {
    margin: 25px 5px 20px 20px;
}

```

ButtonGroup does not have support for **flat** and **round** types.

ButtonGroup styles

The Essential JS 2 ButtonGroup has the following predefined styles. This can be achieved by adding corresponding class name in each button elements.

Class	Description
-------	-------------

- | ----- | ----- |
- | e-primary | Used to represent a primary action. |
- | e-success | Used to represent a positive action. |
- | e-info | Used to represent an informative action. |
- | e-warning | Used to represent an action with caution. |
- | e-danger | Used to represent a negative action. |

The following example illustrates how to achieve predefined styles in ButtonGroup.

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Button-Group</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="e-btn-group">
      <button class="e-btn e-info">View</button>
      <button class="e-btn">Edit</button>
      <button class="e-btn e-danger">Delete</button>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

STYLES.CSS


```
#container {
  visibility: hidden;
}
#loader {
  color: #008cff;
  height: 40px;
  left: 45%;
  position: absolute;
  top: 45%;
  width: 30%;
}
.e-btn-group {
  margin: 25px 5px 20px 20px;
}
```

Predefined ButtonGroup styles provide only the visual indication. So, ButtonGroup content should define the ButtonGroup style for the users of assistive technologies such as screen readers.

See Also

- [ButtonGroup with icons](#)
- [Create ButtonGroup with rounded corner](#)

Selection in ##Platform_Name## Button group control

Selection

Single selection

ButtonGroup supports radio type selection in which only one button can be selected. This can be achieved by adding input element along with `id` attribute with its corresponding label along with `for` attribute inside the target element. In this ButtonGroup, the type of the input element should be `radio` and `e-btn` is added to the `label` element.

The following example illustrates the single selection behavior in ButtonGroup.

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Button-Group</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="e-btn-group">
            <input type="radio" id="radioleft" name="align" value="left">
            <label class="e-btn" for="radioleft">Left</label>
            <input type="radio" id="radiomiddle" name="align"
value="middle">
            <label class="e-btn" for="radiomiddle">Center</label>
            <input type="radio" id="radiatoright" name="align" value="right">
            <label class="e-btn" for="radiatoright">Right</label>
        </div>
    </div>
</body>
</html>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.e-btn-group {
    margin: 25px 5px 20px 20px;
}

```

Multiple selection

ButtonGroup supports checkbox type selection in which multiple button can be selected. This can be achieved by adding input element along with `id` attribute with its corresponding label along with `for` attribute inside the target element. In this ButtonGroup, the type of the input element should be `checkbox` and `e-btn` is added to the `label` element.

The following example illustrates the multiple selection behavior in ButtonGroup.

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>

```

```

<title>EJ2 Button-Group</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="e-btn-group">
            <input type="checkbox" id="checkbold" name="font" value="bold">
            <label class="e-btn" for="checkbold">Bold</label>
            <input type="checkbox" id="checkitalic" name="font"
value="italic">
            <label class="e-btn" for="checkitalic">Italic</label>
            <input type="checkbox" id="checkline" name="font"
value="underline">
            <label class="e-btn" for="checkline">Underline</label>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}

```

```
}  
.e-btn-group {  
  margin: 25px 5px 20px 20px;  
}
```

Nesting

Nesting with other components can be possible in ButtonGroup. The following components can be nested in ButtonGroup.

- DropDownButton
- SplitButton

For nesting support, [SplitButton dependencies](#) should be configured and added in `system.config.js`.

DropDownButton

To initialize DropDownButton component, refer [DropDownButton Getting Started documentation](#).

In the following example, the DropDownButton component can be added by creating button element with ID as `dropdownnelement` in `index.html` and

import the DropDownButton in `script` file, and initialize with the `dropdownnelement`.

INDEX.JS

```
var items = [  
  {  
    text: 'Learn SQL'  
  },  
  {  
    text: 'Learn PHP'  
  },  
  {  
    text: 'Learn Bootstrap'  
  }  
];  
var menuOptions = {  
  items: items  
};  
var btnObj = new ej.splitbuttons.DropDownButton(menuOptions);  
btnObj.appendTo('#dropdownnelement');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>  
  <title>EJ2 Button-Group</title>  
  <meta charset="utf-8">  
  <meta name="viewport" content="width=device-width, initial-scale=1.0">  
  <meta name="description" content="Typescript UI Controls">  
  <meta name="author" content="Syncfusion">  
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">  
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">  
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="e-btn-group">
            <button class="e-btn">HTML</button>
            <button class="e-btn">CSS</button>
            <button class="e-btn">Javascript</button>
            <button class="e-btn" id="dropdownelement">More</button>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.e-btn-group {
    margin: 25px 5px 20px 20px;
}

```

SplitButton

To initialize SplitButton component, refer [SplitButton Getting Started documentation](#).

In the following example, the SplitButton component can be added by creating button element with ID as `splitbuttonelement` in `index.html` and

import the SplitButton in `app.ts` file, and initialize with the `splitbuttonelement`.

INDEX.JS

```

var items = [
{
    text: 'Paste'
},
{
    text: 'Paste Text'
},
{
    text: 'Paste Special'
}
];
var btnObj = new ej.splitbuttons.SplitButton({items: items});
btnObj.appendTo('#splitbuttonelement');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Button-Group</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="e-btn-group">
      <button class="e-btn">Cut</button>
      <button class="e-btn">Copy</button>
      <button class="e-btn" id="splitbuttonelement">Paste</button>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```
#container {
  visibility: hidden;
}
#loader {
  color: #008cff;
  height: 40px;
  left: 45%;
  position: absolute;
  top: 45%;
  width: 30%;
}
.e-btn-group {
  margin: 25px 5px 20px 20px;
}
```

See Also

- [Show ButtonGroup selected state on initial render](#)

Accessibility in ##Platform_Name## Button group control

The Button group component followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Button group component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support | |

| [Section 508](#) Support | |

| Screen Reader Support | |

| Right-To-Left Support | |

| Color Contrast | |

| Mobile Device Support | |

| Keyboard Navigation Support | |

| [Accessibility Checker](#) Validation | |

| [Axe-core](#) Accessibility Validation | |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<div> - All features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>

Keyboard interaction

The Button group component followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Button group component.

Normal behavior

| **Press** | **To do this** |

| --- | --- |

| **Tab** | **Focuses the next button in the ButtonGroup.** |

| **Enter/Space** | **Activates the focussed button in the ButtonGroup.** |

Checkbox behavior

| **Press** | **To do this** |

| --- | --- |

| **Tab** | **Focuses the next button in the ButtonGroup.** |

| **Space** | **Activates the focussed button in the ButtonGroup.** |

Radiobutton behavior

| **Press** | **To do this** |

| --- | --- |

| **Tab** | **Focuses the active button in the ButtonGroup.** |

| **Right** | **Activates next/previous button in the ButtonGroup.** |

Ensuring accessibility

The Button group component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Button group component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Button group component with accessibility tools.

See also

- [Accessibility in Syncfusion ##Platform Name## components](#)

How To

Create buttongroup with icons in ##Platform_Name## Button group control

To create ButtonGroup with icons, `span` element should be added inside each button element with `e-btn-icon` and `e-icon-left` along with icon classes.

The following example illustrates how to create ButtonGroup with icons.

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Button-Group</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="buttongroup" class="e-btn-group">
      <button class="e-btn">
        <span class="e-btn-icon e-icon-left e-icons e-left-
icon"></span>Left
      </button>
      <button class="e-btn">
        <span class="e-btn-icon e-icon-left e-icons e-middle-
icon"></span>Center
      </button>
      <button class="e-btn">
        <span class="e-btn-icon e-icon-left e-icons e-right-
icon"></span>Right
      </button>
    </div>
  </div>
</body>
</html>
```

```

        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.e-btn-group {
    margin: 25px 5px 20px 20px;
}
.e-left-icon::before {
    content: '\e33a';
}
.e-right-icon::before {
    content: '\e34d';
}
.e-middle-icon::before {
    content: '\e35e';
}

```

Create buttongroup with rounded corner in ##Platform_Name## Button group control

The ButtonGroup with rounded corner has round edges on both side. In the ButtonGroup with rounded corner, **e-round-corner** class is to be

added to the target element.

The following example illustrates how to create ButtonGroup with rounded corner.

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Button-Group</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="e-btn-group e-round-corner">
            <button class="e-btn">HTML</button>
            <button class="e-btn">CSS</button>
            <button class="e-btn">Javascript</button>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.e-btn-group {
    margin: 25px 5px 20px 20px;
}

```

Disable in ##Platform_Name## Button group control

Particular button

To disable a particular button in a ButtonGroup, [disabled](#) attribute should be added to the corresponding button element.

Whole ButtonGroup

To disable whole ButtonGroup, [disabled](#) attribute should be added to all the button elements.

The following example illustrates how to disable the particular and the whole ButtonGroup.

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Button-Group</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="e-btn-group">
      <button class="e-btn">HTML</button>
      <button class="e-btn" disabled="">CSS</button>
      <button class="e-btn">Javascript</button>
    </div>
    <div class="e-btn-group">
      <button class="e-btn" disabled="">HTML</button>
      <button class="e-btn" disabled="">CSS</button>
      <button class="e-btn" disabled="">Javascript</button>
    </div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

STYLES.CSS

```
#container {
  visibility: hidden;
}
```

```
#loader {
  color: #008cff;
  height: 40px;
  left: 45%;
  position: absolute;
  top: 45%;
  width: 30%;
}
.e-btn-group {
  margin: 25px 5px 20px 20px;
}
```

To disable radio/checkbox type ButtonGroup, the `disabled` attribute should be added to the particular input element.

Enable ripple in `##Platform_Name##` Button group control

Ripple can be enabled by importing `rippleEffect` method from `ej2-base` and initialize `rippleEffect` with `.e-btn-group` element, and selector as `e-btn`.

The following example illustrates how to enable ripple for ButtonGroup.

<!-- markdownlint-disable MD033 -->

INDEX.JS

```
ej.base.enableRipple(true);
var button = document.querySelector('.e-btn-group');
ej.base.rippleEffect(button, { selector: '.e-btn' });
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Button-Group</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
```

```

        <div id="buttongroup" class="e-btn-group">
            <button class="e-btn">HTML</button>
            <button class="e-btn">CSS</button>
            <button class="e-btn">Javascript</button>
        </div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.e-btn-group {
    margin: 25px 5px 20px 20px;
}

```

Enable rtl in ##Platform_Name## Button group control

ButtonGroup supports RTL functionality. This can be achieved by adding `e-rtl` class to the target element.

The following example illustrates how to create ButtonGroup with RTL support.

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Button-Group</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">
    <link href="styles.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="e-btn-group e-rtl">
            <button class="e-btn">HTML</button>
            <button class="e-btn">CSS</button>
            <button class="e-btn">Javascript</button>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.e-btn-group {
    margin: 25px 5px 20px 20px;
}

```

Form submit in ##Platform_Name## Button group control

The name attribute of the input element is used to group the radio/checkbox type ButtonGroup. When the radio/checkbox type are grouped in the form, the checked items value attribute will be posted to the server on form submit that can be retrieved through the name. The disabled

radio/checkbox type value will not be sent to the server on form submit.

In the following code snippet, the radio type ButtonGroup is explained with male value as checked.

Now, the value that is in checked state will be sent on form submit.

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Button-Group</title>
    <meta charset="utf-8">

```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <form>
            <div class="e-btn-group">
                <input type="radio" id="male" name="gender" value="male"
checked="">
                <label class="e-btn" for="male">Male</label>
                <input type="radio" id="female" name="gender" value="female">
                <label class="e-btn" for="female">Female</label>
                <input type="radio" id="transgender" name="gender"
value="transgender">
                <label class="e-btn" for="transgender">Transgender</label>
            </div>
            <button class="e-btn e-primary">Submit</button>
        </form>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
}

```



```
width: 30%;  
}  
.e-btn-group, button {  
  margin: 20px 5px 20px 20px;  
}
```

Initialize buttongroup using util function in `##Platform_Name##` Button group control

Though, it is a CSS component for easy initialization of ButtonGroup `createButtonGroup` util function can be used.

To use `createButtonGroup` util function, [SplitButton dependencies](#) should be configured and added in `system.config.js` and it should also be imported in `script` file.

Using `createButtonGroup` method, the button options, selector, and `cssClass` is passed and the corresponding classes is added to the elements.

Create basic ButtonGroup

To create a basic ButtonGroup, the target element along with the button elements should be created in `index.html` file.

Create radio type ButtonGroup

To create a radio type ButtonGroup, the target element along with the input elements should be created with type `radio` in `index.html`.

Create checkbox type ButtonGroup

Checkbox type ButtonGroup creation is similar to radio type ButtonGroup, instead the type of the input elements should be `checkbox`.

The following example illustrates how to create ButtonGroup using `createButtonGroup` function for basic, checkbox, and radio

type behaviors.

INDEX.JS

```
ej.splitbuttons.createButtonGroup('#basic', {  
  buttons: [  
    { content: 'HTML' },  
    { content: 'CSS' },  
    { content: 'Javascript' }  
  ]  
});  
ej.splitbuttons.createButtonGroup('#checkbox', {  
  buttons: [  
    { content: 'Bold' },  
    { content: 'Italic' },  
    { content: 'Undeline' }  
  ]  
});  
ej.splitbuttons.createButtonGroup('#radio', {  
  buttons: [  
    { content: 'Left' },  
    { content: 'Center' },  
    { content: 'Right' }  
  ]  
});
```

```
});
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Button-Group</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <h5>Normal behavior</h5>
    <div id="basic">
      <button></button>
      <button></button>
      <button></button>
    </div>
    <h5>Checkbox type behavior</h5>
    <div id="checkboxbox">
      <input type="checkbox" id="checkbold" name="font" value="bold">
      <input type="checkbox" id="checkitalic" name="font"
value="italic">
      <input type="checkbox" id="checkunderline" name="font"
value="underline">
    </div>
    <h5>Radiobutton type behavior</h5>
    <div id="radio">
      <input type="radio" id="radioleft" name="align" value="left">
      <input type="radio" id="radiomiddle" name="align"
value="middle">
      <input type="radio" id="radiatoright" name="align" value="right">
    </div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
```

```

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.e-btn-group {
    margin: 0 5px 5px 5px;
}

```

If null value is passed in button options, then that particular button will be skipped from processing in `createButtonGroup` util function.

Show `buttongroup` selected state on initial render in `##Platform_Name##` Button group control

To show selected state on initial render, `checked` property should be added to the corresponding input element.

The following example illustrates how to show selected state on initial render.

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Button-Group</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

```

```
<div id="container">
  <div class="e-btn-group">
    <input type="checkbox" id="checkbold" name="font" value="bold"
checked="">
    <label class="e-btn" for="checkbold">Bold</label>
    <input type="checkbox" id="checkitalic" name="font"
value="italic">
    <label class="e-btn" for="checkitalic">Italic</label>
    <input type="checkbox" id="checkline" name="font"
value="underline">
    <label class="e-btn" for="checkline">Underline</label>
  </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

STYLES.CSS

```
#container {
  visibility: hidden;
}
#loader {
  color: #008cff;
  height: 40px;
  left: 45%;
  position: absolute;
  top: 45%;
  width: 30%;
}
.e-btn-group {
  margin: 25px 5px 20px 20px;
}
```

Button

Types and styles in ##Platform_Name## Button control

This section explains the different styles and types of Buttons.

Button styles

The Essential JS 2 Button has the following predefined styles that can be defined using the [cssClass](#) property.

Class	Description
-----	-----
e-primary	Used to represent a primary action.

- | e-success | Used to represent a positive action. |
- | e-info | Used to represent an informative action. |
- | e-warning | Used to represent an action with caution. |
- | e-danger | Used to represent a negative action. |
- | e-link | Changes the appearance of the Button like a hyperlink. |

INDEX.JS

```
ej.base.enableRipple(true);
//Primary Button - Used to represent a primary action.
var button = new ej.buttons.Button({ cssClass: `e-primary`, '#primarybtn'});
//Success Button - Used to represent a positive action.
button = new ej.buttons.Button({ cssClass: `e-success`, '#successbtn'});
//Info Button - Used to represent an informative action.
button = new ej.buttons.Button({ cssClass: `e-info`, '#infobtn'});
//Warning Button - Used to represent an action with caution.
button = new ej.buttons.Button({ cssClass: `e-warning`, '#warningbtn'});
//Danger Button - Used to represent a negative action.
button = new ej.buttons.Button({ cssClass: `e-danger`, '#dangerbtn'});
//Link Button - Changes the appearance of the Button like a hyperlink.
button = new ej.buttons.Button({ cssClass: `e-link`, '#linkbtn'});
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Button</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <button id="primarybtn">Primary</button>
    <button id="successbtn">Success</button>
    <button id="infobtn">Info</button>
    <button id="warningbtn">Warning</button>
    <button id="dangerbtn">Danger</button>
    <button id="linkbtn">Link</button>
  </div>
  <script>
    let loader = document.getElementById('loader');
```

```

        let container = document.getElementById('container');
        if (loader) {
            loader.style.display = "none";
        }
        if (container) {
            container.style.visibility = "visible";
        }
    </script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
button {
    margin: 25px 5px 20px 20px;
}

```

Predefined Button styles provide only the visual indication. So, Button content should define the Button style for the users of assistive technologies such as screen readers.

Primary action button can also be achieved by setting [isPrimary](#) property as `true`.

[Button types](#)

The types of Essential JS 2 Button are as follows:

- Basic types
- Flat Button
- Outline Button
- Round Button
- Toggle Button

[Basic types](#)

The basic Button types are explained below.

Type	Description
-----	-----

- | Button | Defines a click Button. |
- | Submit | This Button submits the form data to the server. |
- | Reset | This Button resets all the controls in the form elements to their initial values. |

INDEX.JS

```
ej.base.enableRipple(true);  
var button = new ej.buttons.Button();  
button.appendTo('#submitbtn');  
//Reset.  
button = new ej.buttons.Button();  
button.appendTo('#resetbtn');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>  
  <title>EJ2 Button</title>  
  <meta charset="utf-8">  
  <meta name="viewport" content="width=device-width, initial-scale=1.0">  
  <meta name="description" content="Typescript UI Controls">  
  <meta name="author" content="Syncfusion">  
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">  
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">  
  <link href="styles.css" rel="stylesheet">  
  
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js" type="text/javascript"></script>  
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type="text/javascript"></script>  
</head>  
<body>  
  
  <div id="container">  
    <form>  
      <button type="submit" id="submitbtn">Submit</button>  
      <button type="reset" id="resetbtn">Reset</button>  
    </form>  
  </div>  
  <script>  
    let loader = document.getElementById('loader');  
    let container = document.getElementById('container');  
    if (loader) {  
      loader.style.display = "none";  
    }  
    if (container) {  
      container.style.visibility = "visible";  
    }  
  </script>  
<script>  
var ele = document.getElementById('container');  
if(ele) {  
  ele.style.visibility = "visible";
```

```

}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
button {
    margin: 25px 5px 20px 20px;
}

```

Flat Button

The Flat Button is styled with no background color. To create a flat Button, set the [cssClass](#) property to `e-flat`.

Outline Button

An outline Button has a border with transparent background. To create an outline Button, set the [cssClass](#) property to `e-outline`.

Round Button

A round Button is shaped like a circle. Usually, it contains an icon representing its action. To create a round Button, set the [cssClass](#) property to `e-round`.

INDEX.JS

```

ej.base.enableRipple(true);
var button = new ej.buttons.Button({ cssClass: `e-flat`, '#flatbtn' });
//Outline Button.
button = new ej.buttons.Button({ cssClass: `e-outline`, '#outlinebtn' });
//Round Button - Icon can be loaded by setting "e-icons e-plus-icon" in
"iconCss" property.
//Use "e-icons" class name to load Essential JS 2 built-in icons.
//Use "e-plus-icon" class name to load plus icon that you can refer in
"styles.css".
button = new ej.buttons.Button({ cssClass: `e-round`, iconCss: 'e-icons e-
plus-icon', isPrimary:true }, '#roundbtn');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Button</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

```



```

<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="flatbtn">Flat</button>
        <button id="outlinebtn">Outline</button>
        <button id="roundbtn"></button>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
/* For Round Button*/
.e-plus-icon::before {
    content: '\e823';
}
button {
    margin: 25px 5px 20px 20px;
}

```

Toggle Button

A toggle Button allows you to change between the two states. The Button is active in toggled state and can be recognized through the `e-active` class. The functionality of the toggle Button is handled by click

event. To create a toggle Button, set the [isToggle](#) property to `true`. In the following code snippet, the toggle Button text changes to play/pause based on the state of the Button with the use of click event.

INDEX.JS

```
ej.base.enableRipple(true);
var togglebtn = new ej.buttons.Button({cssClass: `e-flat`, iconCss: 'e-btn-sb-icon e-play-icon', isToggle: true, content: 'Play'}, '#togglebtn');
//Click Event.
document.getElementById('togglebtn').onclick = function () {
    if (document.getElementById('togglebtn').classList.contains('e-active')) {
        togglebtn.content = 'Pause';
        togglebtn.iconCss = 'e-btn-sb-icon e-pause-icon';
    } else {
        togglebtn.content = 'Play';
        togglebtn.iconCss = 'e-btn-sb-icon e-play-icon';
    }
};
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Button</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <button id="togglebtn"></button>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
@font-face {
    font-family: 'btn-icon';
    src:
        url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMjltSfgAAAEoAAAAVmNtYXDNH+dzAAABoAAAAEJnbHlmlv4
8pAAAAfgAAAQYagVhZBOPfZcAAADQAAAAANmhoZWEIUQQJAAAArAAAACRobXR4IAAAAAAYAAAAA
gbG9jYQON6ApQAAAHkAAAAEm1heHABFQCqAAABCAAAACBuYW1l071FxAABhAAAAIxcG9zdK9uovo
AAAhEAAAAGaABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAACAAABAAAAQAAJ1LUzF8
PPPUACwQAAAAAANg+nFMAAAAA2D6cUwAAAAAD9AP0AAAACAACAAAAAAAAAAAAEAAAAIAJ4AAwAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQQAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnBgQAAAAAXAQAAAAAAAAABAAAAAAAAABAAAAAQAAA
EAAAABAAAAAQAAAAEAAAABAAAAAQAAAAAACAAAAwAAABQAAwABAAAAFAAEAC4AAAAEAAQAAQA
A5wb//wAA5wD//wAAAAEABAAAAAEAAgADAAQABQAGAAcAAAAAAAAAADgAkADIaHAeUAewCDAAAAAE
AAAAAA2ED9AACAAA3CQGeAsT9PAwB9AH0AAACAAAAAAPHa/QAAwAHAAALIREhASERIQJpAV7+ov3
QAV7+ogwD6PwYA+gAAAEAAAAAA4sD9AACAAATARF0AxgCAP4MA+gAAAAABAAAAAAP0A/QAQwAAExE
fDyE/DxEvDyEPDgWBagMFBQcICQkLCwMDQ4NatoNDg0MDAsLCQkIBwUFAwIBAQIDBQUHCAkJCws
MDA0ODf0mDQ4NDawLCwkJCACFBQMCA239Jg4NDQ0LCwsJCQgHBQUDAgEBAgMFBQcICQkLCwsNDQ0
OAtOoDQ0NCwsLCQkIBwUFAwIBAQIDBQUHCAkJCwsLDQ0NAAIAAAAAA/MDxQADAIwAADczESMBDwM
VFw8METM3HwQ3Fz8KPQEvBT8LLwg3NT8INS8FNT8NNS8JByU/BDUvCyMPAQytrQH5AgoEAQEBAg
hERESEyIJSgQBiEHNQceOZPbDgUICw0LCQUDBAICBAkGAgEBAQMOBAkIBgcDAwEBAQEDAwMJAQE
BAxYLBQQEAwMCAGIEBAoBAQEECgcHBgUFBAMDAQEBAQQFBwkFBQUGEf6tDwkEawIBAQMDCgwVawc
GDAsNBwdaAYcB3gEFAwN2HwoELDodGxwaLwkIGwz+igEBHwMBAQECAQEDBgoKDAYICAgFCAkICwU
EBAQFAwYDBwgIDAgHCACGBgYFBQkEAgYCBawJBgUGBwkJCgkICAcLBAIFAwIEBAQFBQcGBwgHBgY
GBgoJCAYCAGeBAQFGMRkaGw0NDA0LIh4xBAQCBAEBAGADAAAAAOKA/MAHABCAJ0AAAEzHwIRDwM
hLwIDNzM/CjUTHwcVIwcVIy8HETcXmz8KNScxBxEfdjsBHQEfdTMhMz8OES8PIz0BLw4hA0EDBQQ
DAQIEBf5eBQQCAW4RDg0LCQgGBQUDBAFEBAMDawIBAQGL7Y0EawQCAgIBAYYKChEQDQsJCACeBAU
CYt8BAQIDBAUFBQcHBwgICQgKjQECAGMEBAUFBgYHBgcIBwGcCAcHBwYGBgUFBAQDAgIBAQEBAgI
DBAQFBQYGBgcHBwgMAQMDAwUFBgYHBwgICQkJ/tQCiwMEBf3XAwYEAgIEBgFoAQEDBQYGBwgIBw0
KhQEiAQEBAgMDAwTV+94BAQECAwMDBAGyAQECBAYHCAgJCgkQCaQC6/47CQkICQcIBwYGBQQEAwI
CUAgHBwcGBgYFBQQAeAwMBAgIBAwMEBAUFBQcGBwCHCAImCAcHBwYGBgUFBAQDAgIBAdUJCQgICAg
GBwYFBAQDAgEBAAAAAIAAAAAA6cD9AADAaAAdchNSElAQcJAScBESNZA078sgGB/uMuAXkBgDb
+1EwMTZcBCD3+ngFiPf7pAxMAAAAAABIA3gABAAAAAIAAAAAAABAAAAAABAAgAAQABAAAAAA
CAACACQABAAAAAADAAGAEAAABAAAAAEEAGAGAABAAAAAFAAASAIABAAAAAAGAAGAKwABAAA
AAAAKACwAMwABAAAAAALABIAXwADAAEECQAAAAIAcQADAAEECQABABAAcWADAAEECQACAA4AgwA
DAAEECQADABAAkQADAAEECQAEABAAoQADAAEECQAFABYAsQADAAEECQAGABAAxwADAAEECQAKAFg
AlwADAAEECQALACQBLyBidG4taWNvblJlZ3VsYXJidG4taWNvbmJ0bilpY29uVmVyc2lvbiAxLjB
idG4taWNvbkZvbnQgZ2VuZXJhdGVkIHVzaW5nIFN5bmNmdXNpb24gTWV0cm8gU3RlZGlvd3d3LnN
5bmNmdXNpb24uY29tACAAYgB0AG4ALQBPAGMabwBuAFIAZQBnAHUAbABhAHIAyAgB0AG4ALQBPAGM
AbwBuAGIAAdABuAC0AAQBJAG8ABgBWAGUAcgBzAGkAbwBuACAAMQAuADAAYgB0AG4ALQBPAGMabwB
uAEYAbwBuAHQAIAbnAGUAbgBlAHIAyAgB0AGUAZAAGAHUAcwBpAG4AZwAgAFMAeQBwAGMAZgB1AHM
AaQBvAG4AIAABNAGUAdABYAG8AIAABTAHQAdQBkAGkAbwB3AHcAdwAuAHMAeQBwAGMAZgB1AHMAaQB
vAG4ALgBjAG8ABQAAAAAIAAAAAAIAAAAAAIAAAAAAIAAAAAAIAAAAAAIAAAAAAIAAAAAAIAAAAA
GAQcBCAEJAAPtZWRpYS1wbGF5C211ZG1hLXBhdXNlDmFycm93aGVhZC1sZWZ0BHN0b3AJbGlrZS0
tLTaxBGNvcHkQLWRvd25sb2FkLTAYLXdmlQAA) format('true-type');
    font-weight: normal;
    font-style: normal;
}

```

```

}
.e-btn-sb-icon {
    font-family: 'btn-icon' !important;
    speak: none;
    font-size: 55px;
    font-style: normal;
    font-weight: normal;
    font-variant: normal;
    text-transform: none;
    line-height: 1;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
}
/* Added when toggle button is in active state*/
.e-play-icon::before {
    content: '\e700';
}
/* Added when toggle button is not in active state*/
.e-pause-icon::before {
    content: '\e701';
}

```

Icons

Button with font icons

The Button can have an icon to provide the visual representation of the action. To place the icon on a Button, set the [iconCss](#) property with the required icon CSS. By default, the icon is positioned to the left side of the Button. You can customize the icon's position by using the [iconPosition](#) property.

INDEX.JS

```

ej.base.enableRipple(true);
//To position the icon to the left of the text on a Button.
var button = new ej.buttons.Button({iconCss: 'e-btn-sb-icon e-prev-icon'},
'#iconbutton');
//To position the icon to the right of the text on a Button.
button = new ej.buttons.Button({iconCss: 'e-btn-sb-icon e-stop-icon',
iconPosition: 'Right'}, '#iconposbtn');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Button</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>

```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="iconbutton">Previous</button>
        <button id="iconposbtn">Stop</button>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
@font-face {
    font-family: 'btn-icon';
    src:
        url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMjltSfgAAAEoAAAAVmNtYXNlH+dzAAABoAAAAEJnbHlm1v4
8pAAAAfgAAAQYagVhZBOPfZcAAADQAAAAANmhoZWEIUQQJAAAArAAAACRobXR4IAAAAAAYAAAAA
gbG9jYQYQAAAHkAAAAEm1heHABFQCqAAABCAAAACBuYw1l07lFxAAABhAAAAIxcG9zdK9uovo
AAAhEAAAAGaABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAAABAAAAAAQAAJ1LUzF8
PPUACwQAAAAAANg+nFMAAAAA2D6cUwAAAAAD9AP0AAAAACAACAAAAAAAAAAAAEAAAAIAJ4AAwAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnBgQAAAAAXAQAAAAAAAAABAAAAAAAABAAAAAQAAA
EAAAAABAAAAAQAAAAEAAAAABAAAAAQAAAAAAACAAAAAwAAABQAAwABAAAAFAAEAC4AAAAEAAQAAQA
A5wb//wAA5wD//wAAAAEABAAAAEAAgADAAQABQAGAAcAAAAAAAAADgAkADIAhAEuAewCDAAAAAE
AAAAAA2ED9AACAAA3CQGeAsT9PAwB9AH0AAACAAAAAAPHa/QAAwAHAAALIREhASERIQJpAV7+ov3
QAV7+ogwD6PwYA+gAAAEAAAAAA4sD9AACAAATARF0AxgCAP4MA+gAAAAABAAAAAAP0A/QAQwAAExE
fDyE/DxEvDyEPDgwBAGMFBQcICQkLCwwMDQ4NAtONDg0MDAsLCQkIBwUFAwIBAQIDBQUHCAkJCws
MDA00df0mDQ4NDawLCwkJCACFBQMCA239Jg4NDQ0LCwsJCQgHBQUdAgEBAgMFBQcICQkLCwsNDQ0
OAtOoDQ0NCwsLCQkIBwUFAwIBAQIDBQUHCAkJCwsLDQ0NAAIAAAAAA/MDxQADAIwAADczESMBDwM
VFw8METM3HwQ3Fz8KPQEVBt8LLwg3NT8INS8FNT8NNS8JBjYU/BDUVCyMPAQytrQH5AgoEAQEBArg
hERESEyIJCSgQBIEHNQceOZPbDgUICw0LCQUDBAICBAkGAgEBAQMOBAkIBgcDAwEBAQEDAwMJAge
BAxYLBQQEAWMCAgIEBAoBAQEECgcHBgUFBAMDAQEBAQQFBwkFBQUGef6tDwKEAwIBAQMDCgwVAwC
GDAsNBwdaAYcB3gEFAwN2HwoELDodGxwaLwkIGwz+igEBHwMBAQECAQEDBgoKDAYICAgFCAkICwU
EBAQFAwYDBwgIDAgHCACGBgYFBQkEAgYCBawJBgUGBwkJCgkICAcLBAIFAwIEBAQFBQcGBwgHBgY
GBgoJCAYCAGeBAQFGMRkaGw0NDA0LIh4xBAQCBAEBAGADAAAAAOKA/MAHABCAJ0AAAEzHwIRDwM
hLwIDNzM/CjUTHwcVIwcVIy8HETcXMz8KNScxBxEfDjsSBHQEfDTMhMz8OES8PiZ0BLw4hA0EDBQQ

```

```

DAQIEBf5eBQQCAW4RDg0LCQgGBQUDBAFEBAMDawIBAQL7Y0EAWQCAgIBAYYKChEQDQsJCAcEBAU
CYt8BAQIDBAUFbQcHBwgICQgKjQECAGMEBAUFbgYHBgcIBwGcCAcHBwYGBgUFBAQDAgIBAQBAGI
DBAQFBQYGBgcHBwgmAQMDAwUFbgYHBwgICQkJ/tQCiwMEBf3XAwYEAgIEBgFoAQEDBQYGBwgIBw0
KhQEiAQEBAGMDAwTV+94BAQECaWMDBAGyAQECBAYHCAgJCgkQCaQC6/47CQkICQcIBwYGBQQEAwI
CUAgHBwcGBgYFBQQEAwMBAgIBAwMEBAUFbQcGBwcHCAImCAcHBwYGBgUFBAQDAgIBAdUJCQgICAg
GBwYFBAQDAgEBAAAAAIAAAAAA6cD9AADAaWaadchNSElAQcJAScBESNZAO78sgGB/uMuAXkBgDb
+1EwMTZcBCD3+ngFiPf7pAxMAAAAAABIA3gABAAAAAEEEEAAAAAABAAAAAABAAgAAQABAAAAA
CAACACQABAAAAAADAAGAEAAABAAAAAEEAGAGAABAAAAAFAAsAIAABAAAAAAGAAGAKwABAAA
AAAAKACwAMwABAAAAAALABIAXwADAAEEECQAAAAIACQADAAEEECQABABAAcwADAAEEECQACAA4AgwA
DAAEEECQADABAAKQADAAEEECQAEABAAoQADAAEEECQAFABYAsQADAAEEECQAGABAAxwADAAEEECQAKAFg
AlwADAAEEECQALACQBLyBidG4taWNvblJlZ3VsYXJidG4taWNvbmJ0bilpY29uVmVyc2lubiAxLjB
idG4taWNvbkbZvbnQgZ2VuZXJhdGVkIHVzaW5nIFN5bmNmdXNpb24gTWV0cm8gU3RlZGlvd3d3LnN
5bmNmdXNpb24uY29tACAAYgB0AG4ALQBpAGMABwBuAFIAZQBnAHUAbABhAHIAAYgB0AG4ALQBpAGM
AbwBuAGIAAdABuAC0AaQBjAG8AbgBWAGUAcgBzAGkAbwBuACAAMQAuADAAYgB0AG4ALQBpAGMABwB
uAEYAbwBuAHQAIAbnAGUAbgBlAHIAAYQB0AGUAZAAGAHUAcwBpAG4AZwAgAFMAeQBwAGMAZgBlAHM
AaQBvAG4AIAbnAGUAdABYAG8AIAbTAHQAdQBkAGkAbwB3AHcAdwAuAHMAeQBwAGMAZgBlAHMAaQB
vAG4ALgBjAG8AbQAAAAAIAAAAAAaAAAAAaAAAAAaAAAAAaAAAAAaAAAAAaAAAAAaAAAAAaAAAAAa
GAQcBCAEJAAPtZWRpYS1wbGF5C2l1ZGlhLXBhdXNlDmFycm93aGVhZC1sZWZ0BHN0b3AJbGlrZS0
tLTAXBGNvcHkQLWRvd25sb2FkLTAYLXdmlQAA) format('true');
    font-weight: normal;
    font-style: normal;
}
.e-btn-sb-icon {
    font-family: 'btn-icon' !important;
    speak: none;
    font-size: 55px;
    font-style: normal;
    font-weight: normal;
    font-variant: normal;
    text-transform: none;
    line-height: 1;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
}
/*For Right Icon Button*/
.e-stop-icon::before {
    content: '\e703';
}
/*For Left Icon Button*/
.e-prev-icon::before {
    content: '\e702';
}
button {
    margin: 25px 5px 20px 20px;
}

```

Button with SVG image

SVG image can be added to the Button using [iconCss](#) property.

In the following example, SVG image is added using the iconCss class `e-search-icon` by setting `height` and `width`.

INDEX.JS

```
ej.base.enableRipple(true);
```

```
var button = new ej.buttons.Button({ iconCss: 'e-search-icon' },  
'#iconbutton');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>  
  <title>EJ2 Button</title>  
  <meta charset="utf-8">  
  <meta name="viewport" content="width=device-width, initial-scale=1.0">  
  <meta name="description" content="Typescript UI Controls">  
  <meta name="author" content="Syncfusion">  
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-  
base/styles/material.css" rel="stylesheet">  
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-  
buttons/styles/material.css" rel="stylesheet">  
  <link href="styles.css" rel="stylesheet">  
  
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"  
type="text/javascript"></script>  
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type  
="text/javascript"></script>  
</head>  
<body>  
  
  <div id="container">  
    <button id="iconbutton"></button>  
  </div>  
  <script>  
var ele = document.getElementById('container');  
if(ele) {  
  ele.style.visibility = "visible";  
}  
  </script>  
  <script src="index.js" type="text/javascript"></script>  
</body></html>
```

STYLES.CSS

```
#container {  
  visibility: hidden;  
}  
#loader {  
  color: #008cff;  
  height: 40px;  
  left: 45%;  
  position: absolute;  
  top: 45%;  
  width: 30%;  
}  
.e-btn-icon.e-search-icon {  
  background: url('search_icon.svg');  
  height: 25px;  
  width: 25px;  
}  
button {
```

```
margin: 25px 5px 20px 20px;
}
```

The Essential JS 2 provides a set of icons that can be loaded by applying `e-icons` class name to the element. You can also use third party icons on the Button using the [iconCss](#) property.

Button size

The two types of Button sizes are default and small. To change the size of the default Button to small Button, set the [cssClass](#) property to `e-small`.

INDEX.JS

```
ej.base.enableRipple(true);
//Small Button.
var button = new ej.buttons.Button({ cssClass: `e-small`, '#smallbtn' });
//Normal Button.
button = new ej.buttons.Button();
button.appendTo('#normalbtn');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Button</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <button id="smallbtn">Small</button>
    <button id="normalbtn">Normal</button>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```


STYLES.CSS

```
#container {
  visibility: hidden;
}
#loader {
  color: #008cff;
  height: 40px;
  left: 45%;
  position: absolute;
  top: 45%;
  width: 30%;
}
button {
  margin: 25px 5px 20px 20px;
}
```

See Also

- [Customize Button appearance](#)
- [How to create block button](#)
- [How to create repeat button](#)

Accessibility in ##Platform_Name## Button component

The Button component followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Button component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support | |

| [Section 508](#) Support | |

| Screen Reader Support | |

| Right-To-Left Support | |

| Color Contrast | |

| Mobile Device Support | |

| Keyboard Navigation Support | |

| [Accessibility Checker](#) Validation | |

| [Axe-core](#) Accessibility Validation | ` |`

`<style>`

`.post .post-content img {`

`display: inline-block;`

`margin: 0.5em 0;`

`}`

`</style>`

`<div> - All features of the component meet the requirement.</div>`

`<div> - Some features of the component do not meet the requirement.</div>`

`<div> - The component does not meet the requirement.</div>`

WAI-ARIA attributes

The Button component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Button component:

| Attributes | Purpose |

| --- | --- |

| **aria-label** | Provides an accessible name for the icon only button. |

Keyboard interaction

The Button component followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Button component.

| **Press** | **To do this** |

| --- | --- |

| **Space** | When the button has focus, pressing the space key changes the state of the button. |

Ensuring accessibility

The Button component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Button component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Button component with accessibility tools.

See also

- [Accessibility in Syncfusion ##Platform_Name## components](#)

How To

Add link to a button in ##Platform_Name## Button control

The appearance of the Button can be changed like a link by `e-link` class using `cssClass` property and link navigation can be handled in Button click.

In the following example, link is added in Button click by using `window.open()` method.

INDEX.JS

```
ej.base.enableRipple(true);
//To change the Button type.
var button = new ej.buttons.Button({cssClass: 'e-link'});
button.appendTo('#element');
button.element.onclick = function () {
    window.open("https://www.google.com");
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Button</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <button id="element">Go to Google</button>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

STYLES.CSS

```
#container {
  visibility: hidden;
```

```

}
#loader {
  color: #008cff;
  height: 40px;
  left: 45%;
  position: absolute;
  top: 45%;
  width: 30%;
}

```

Create a block button in ##Platform_Name## Button control

You can customize a Button into a Block Button that will span the entire width of its parent element. To create a Block Button, set the [cssClass](#) property to `e-block`.

INDEX.JS

```

ej.base.enableRipple(true);
//Block Button.
var button = new ej.buttons.Button({cssClass: `e-block`});
button.appendTo('#blockbtn');
//Primary Block Button.
button = new ej.buttons.Button({ isPrimary: true, cssClass: `e-block` });
button.appendTo('#primarybtn');
//Success Block Button.
button = new ej.buttons.Button({ cssClass: `e-block e-success` });
button.appendTo('#successbtn');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Button</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <button id="blockbtn">Block Button</button>
    <button id="primarybtn">Block Button</button>
    <button id="successbtn">Block Button</button>
  </div>
  <script>

```

```

        let loader = document.getElementById('loader');
        let container = document.getElementById('container');
        if (loader) {
            loader.style.display = "none";
        }
        if (container) {
            container.style.visibility = "visible";
        }
    }
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLE.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
button {
    margin: 25px 0;
}

```

Customize button appearance in ##Platform_Name## Button control

You can customize the appearance of the Button by using the Cascading Style Sheets (CSS). Define the CSS according to your requirement, and assign the class name to the [cssClass](#) property. In the following code snippet the background color, text color, height, width, and sharp corner of the Button can be customized through the `e-custom` class for all states (hover, focus, and active).

INDEX.JS

```

ej.base.enableRipple(true);
var button = new ej.buttons.Button({cssClass: `e-custom`, content: 'Custom'},
'#custombtn');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Button</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">

```

```

    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="styles.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="custombtn"></button>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
/* To customize button appearance */
.e-custom {
    border-radius: 0;
    height: 30px;
    width: 80px;
}
.e-left-icon::before {
    content: '\e7d4';
}
.e-right-icon::before {
    content: '\e916';
}
.e-custom, .e-custom:hover, .e-custom:focus, .e-custom:active {
    background-color: #ff6e40;
    color: #fff;
}

```

```

button {
  margin: 25px 5px 20px 20px;
}
@font-face {
  font-family: 'settings';
  src:
    url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMj0gSRsAAAEoAAAAVmNtYXNlEODVAAABiAAAADZnbHlm7/n
pHAAAAcGAAAEwaGVhZBktDIMAAADQAAAAANmhoZWEHmQNrAAAArAAAACRobXR4B+gAAAAAYAAAAA
IbG9jYQCYAAAAAHAAAAABmlheHABDgB0AAABCAAAACBuYWl1xmFdywAAAvGAAAIxcG9zdDwSCic
AAAUAAAAANwABAAADUv9qAFoEAAAA//4D6gABAAAAAAAAAAAAAAAAAAAgABAAAAQAAMLNyX18
PPPUACwPoAAAAANfSY9oAAAAA19Jj2gAAAAAD6gPqAAAAACAACAAAAAAAAAAAEAAAAACAGgAAgAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQP0AZAABQAAAnoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnAANS/2oAWgPqAJYAAAAABAAAAAAAAABAAAAAPoAAA
AAAACAAAAAAwAAABQAAwABAAAAFAAEACIAAAAEAAQAAQAA5wD//wAA5wD//wAAAAEABAAAAAEAAAA
AAAAmAAAAAIAAAAAA+oD6gALAGCAAAEOAQcuASc+ATceAQEVBgcNjiIPAQYUHWEOAQcjIgYdAR4
BOwEWFwcGFB8BFjI/AR4BFxUeATsBPgE9ATY3FxYyPwE2NC8BPgE3MzI2PQE0JisBJic3NjQvASY
iDwEuASc1LgEnIw4Bar4CcVVUCQMDcVRVcf7pOTRbBRQGUQYGWhAYB30LDgEPCX0OIVoHB1EFFQV
bGTyEaQ4KcQsOOTRbBRQGUQYGWhAXB34LDQ8Jfg4gWgcHUQUVBVsZNh4BDgpxCQ8B9lRxAwNxFV
xAgJxAyd+DiBaBgZRBRUFWBk2Hg8KcQsOOTRbBxQHUYGWhAYB30LDgEPCX0OIVoGBk4FFQVbGTy
eDwpxCw06NFoIEwhRBgZaEBgHfQsNAQENAAAAABIA3gABAAAAAAAAAAAEAAAABAAAAAABAAgAAQA
BAAAAAAACAACACQABAAAAAADAAGAEAAABAAAAAAAEAAgAGAABAAAAAAFAAAsAIAABAAAAAAGAAG
AKwABAAAAAAAKACwAMwABAAAAAALABIAXwADAAEECQAAAAIACQADAAEECQABABAACwADAAEECQA
CAA4AgwADAAEECQADABAAKQADAAEECQAEABAAoQADAAEECQAFABYAsQADAAEECQAGABAAxwADAAE
ECQAKAFgAlwADAAEECQALACQBLyBzZXR0aW5nc1JlZ3VsYXJzZXR0aW5nc3NldHRpbmdzVmVyc2l
vbiAxLjBzZXR0aW5nc0ZvbnQgZ2VuZXJhdGVkIHVzaW5nIFN5bmNmdXNpb24gTWV0cm8gU3RlZG1
vd3d3LnN5bmNmdXNpb24uY29tACAACwBLAHQAdABpAG4AZwBzAFIAZQBnAHUAbABhAHIAcWBLAHQ
AdABpAG4AZwBzAHMAZQB0AHQAaQBwAGcAcwBWAGUAcgBzAGkAbwBuACAAMQAuADAACwBLAHQAdAB
pAG4AZwBzAEYAbwBuAHQAIAABnAGUAbgBLAHIAAYQB0AGUAZAAGAHUAACwBpAG4AZwAgAFMAEQBuAGM
AZgB1AHMAaQBvAG4AIAABNAGUAdABYAG8AIAABTAHQAdQBkAGkAbwB3AHcAdwAuAHMAEQBuAGMAZgB
1AHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAAAAAAoAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAIBAgE
DAA1zZXR0aW5ncy0tLkExAAAA) format('trueType');
  font-weight: normal;
  font-style: normal;
}
.e-btn-icons {
  font-family: 'settings' !important;
  speak: none;
  font-size: 55px;
  font-style: normal;
  font-weight: normal;
  font-variant: normal;
  text-transform: none;
  line-height: 1;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}
.e-setting-icon::before {
  content: "\e700";
}

```

Customize input and anchor elements in ##Platform_Name## Button control

You can customize the appearance of the input and anchor elements using predefined styles through the class property. In the following code

snippet, the input element is customized as a link Button by setting the `e-btn e-link` class, and the anchor element is customized as a

primary Button by setting the `e-btn e-primary` class.

INDEX.TS

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Button</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div>
    <input type="button" id="inputbtn" value="Input Button" class="e-btn
e-link">
  </div><br>
  <div>
    <a id="anchorbtn" class="e-btn e-primary" href="#">Google</a>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Repeat button in ##Platform_Name## Button control

The Repeat button is a type of Button in that the click event is triggered at regular time interval from the pressed state till the released state.

The following example explains about how to achieve Repeat Button in mouse and touch events.

INDEX.JS

```
ej.base.enableRipple(true);
var button = new ej.buttons.Button({cssClass: `e-small`}, '#button');
```



```

var clear = new ej.buttons.Button({cssClass: `e-small`, '#clear'});
var timeout = null;
document.getElementById('clear').onclick = () => {
    document.getElementById('eventlog').innerHTML = '';
};
document.getElementById('button').addEventListener("mousedown",
mousedownHandler);
document.getElementById('button').addEventListener("mouseup",
mouseupHandler);
function mouseUpHandler(event) {
    clearInterval(timeout);
}
function mousedownHandler(event) {
    if (event.which === 1) {
        appendSpanElement('Button click event called<hr>');
        timeout = setInterval(clickEventHandler, 200);
    }
}
function clickEventHandler() {
    ej.base.EventHandler.trigger(button.element, "click");
    appendSpanElement('Button click event called<hr>');
}
function appendSpanElement(text) {
    var span = document.createElement('span');
    span.innerHTML = text;
    var log = document.getElementById('eventlog');
    log.insertBefore(span, log.firstChild);
}
document.getElementById('loader').style.display = "none";
document.getElementById('container').style.visibility = "visible";

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Button</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="styles.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="btncontainer">
            <button id="button">Button</button>

```

```

        </div>
        <div class="event" style="height:auto;">
            <table title="Event Trace" style="width:100%">
                <tbody>
                    <tr>
                        <th>Event Trace</th>
                    </tr>
                    <tr>
                        <td>
                            <div class="eventarea" style="height: 250px;overflow:
auto">
                                <span id="eventlog" style="word-break:
normal;"></span>
                            </div>
                        </td>
                    </tr>
                    <tr>
                        <td>
                            <div class="evtbtn" style="padding:20px 0 0 20px">
                                <button id="clear">Clear</button>
                            </div>
                        </td>
                    </tr>
                </tbody>
            </table>
        </div>
    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.btncontainer {
    float: left;
    width: 40%;
}
.event {
    float: right;
    width: 60%;
}

```

```
border-left: 1px solid #D7D7D7;
}
#eventlog b {
  color: #388e3c;
}

hr {
  margin: 1px 10px 1px 0px;
  border-top: 1px solid #eee;
}
```

Right to left in ##Platform_Name## Button control

Button component has RTL support. This can be achieved by setting [enableRtl](#) as `true`.

The following example illustrates how to enable right-to-left support in Button component.

INDEX.JS

```
ej.base.enableRipple(true);
var button = new ej.buttons.Button({iconCss: 'e-btn-icons e-setting-icon',
content: 'Settings', enableRtl: true}, '#custombtn');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Button</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <button id="custombtn"></button>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
/* To customize button appearance */
.e-custom {
    border-radius: 0;
    height: 30px;
    width: 80px;
}
.e-left-icon::before {
    content: '\e7d4';
}
.e-right-icon::before {
    content: '\e916';
}
.e-custom, .e-custom:hover, .e-custom:focus, .e-custom:active {
    background-color: #ff6e40;
    color: #fff;
}
button {
    margin: 25px 5px 20px 20px;
}
@font-face {
    font-family: 'settings';
    src:
    url(data:application/x-font-ttf;charset=utf-
    8;base64,AAEAAAAKAIAAAwAgTlMvMj0gSRsAAAEoAAAAVmNtYXDNEdVAAABiAAAADZnbHlm7/n
    pHAAAACgAAAEwaGVhZBKTdIMAADQAAANmhoZWEHmQNrAAAArAAAACRobXR4B+gAAAAAYAAAA
    IbG9jYQCYAAAAAHAAAAABmlheHABDgB0AAABCAAAACBuYW11xmFdywAAAvGAAAIxcG9zdDwSCic
    AAUsAAAAANwABAAADUv9qAFoEAAAA//4D6gABAAAAAAAAAAAAAAAAAAgABAAAAQAAMLNyX18
    PPUACwPoAAAAANfSY9oAAAAA19Jj2gAAAAAD6gPqAAAAACAACAAAAAAAAAAAEAAAACAGgAAgAAAA
    AAgAAAAoACgAAAP8AAAAAAAAAAQP0AZAABQAAAnoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
    AAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnAANS/2oAWgPqAJYAAAABAAAAAAAAABAAAAAPoAAA
    AAAACAAAAAwAAABQAAwABAAAAFAAEACIAAAAEAAQAAQAA5wD//wAA5wD//wAAAAEABAAAAAEAAAA
    AAAAAmAAAAAIAAAAAA+oD6gALAGcAAAEoAQcuASc+ATceAQEVBgcnJiIPAQYUHWEOAQcjIgYdAR4
    BOWEWFwcGFB8BFjI/AR4BFxUeATsBPgE9ATY3FxYYPwE2NC8BPgE3MzI2PQE0JisBJic3NjQvASY
    iDwEuASc1LgEnIw4BAr4CcVVUcQMDcVRVcf7pOTRbBRQGUQYgWhAYB30LDgEPCX00IVoHB1EFFQV
    bGTyEAQ4KcQsOOTRbBRQGUQYgWhAXB34LDQ8JfG4gWgcHUQUVBVszNh4BDgpxCQ8B9lRxAwNxVfV
    xAgJxAYd+DiBaBgZRBRUFWBk2Hg8KcQsOOTRbBxQHUYGWhAYB30LDgEPCX00IVoGBk4FFQVbGTy
    eDwpxCw06NfoIEwhRBgZaEBgHfQsNAQENAAAAABIA3gABAAAAAAAAAAAEAAAABAAAAAABAAGAAQA
    BAAAAAACAAcACQABAAAAAADAAgAEAAABAAAAAAAEAAgAGAABAAAAAAFAAAsAIAABAAAAAAGAAG
    AKwABAAAAAAAKACwAMwABAAAAAALABIAXwADAAEECQAAAAIACQADAAEECQABABAAcwADAAEECQA
    CAA4AgwADAAEECQADABAAkQADAAEECQAEABAAoQADAAEECQAFABYAsQADAAEECQAGABAAxwADAAE
    ECQAKAFgAlwADAAEECQALACQBLyBzZXR0aW5nc1JlZ3VsYXJzZXR0aW5nc3NldHRpbmdzVmVyc2l
    vbiAxLjBzZXR0aW5nc0ZvbG9zZ2VuZXJhdGVkIHVzaW5nIFN5bmNmdXNpb24gTWV0cm8gU3R1ZG1
    vd3d3LnN5bmNmdXNpb24uY29tACAacwBIAHQAdABpAG4AZwBzAFIAZQBnAHUAbABhAHIAcwBIAHQ
    AdABpAG4AZwBzAHMAZQB0AHQAaQBuAGcAcwBWAGUAcgBzAGkAbwBuACAAMQAuADAAcwBIAHQAdAB
    
```

```

pAG4AZwBzAEYAbwBuAHQAIABnAGUAbgBlAHIAyQB0AGUAZAAGAHUAcwBpAG4AZwAgAFMAeQBuAGM
AZgBlAHMAaQBvAG4AIABNAGUAdABYAG8AIABTAHQAdQBkAGkAbwB3AHcAdwAuAHMAeQBuAGMAZgB
lAHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAAAAAAoAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAIBAgE
DAA1zZXR0aW5ncy0tLTExAAAA) format('true');
font-weight: normal;
font-style: normal;
}
.e-btn-icons {
font-family: 'settings' !important;
speak: none;
font-size: 55px;
font-style: normal;
font-weight: normal;
font-variant: normal;
text-transform: none;
line-height: 1;
-webkit-font-smoothing: antialiased;
-moz-osx-font-smoothing: grayscale;
}
.e-setting-icon::before {
content: "\e700";
}

```

Set the disabled state in `##Platform_Name##` Button control

Button component can be enabled/disabled by giving [disabled](#) property. To disable Button component, the `disabled` property can be set as `true`.

The following example demonstrates Button in `disabled` state.

INDEX.JS

```

ej.base.enableRipple(true);
var button = new ej.buttons.Button({content: 'Disabled', disabled: true},
'#custombtn');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Button</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>

```

```
<body>

  <div id="container">
    <button id="custombtn"></button>
  </div>

<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

STYLES.CSS

```
#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
/* To customize button appearance */
.e-custom {
    border-radius: 0;
    height: 30px;
    width: 80px;
}
.e-left-icon::before {
    content: '\e7d4';
}
.e-right-icon::before {
    content: '\e916';
}
.e-custom, .e-custom:hover, .e-custom:focus, .e-custom:active {
    background-color: #ff6e40;
    color: #fff;
}
button {
    margin: 25px 5px 20px 20px;
}
@font-face {
    font-family: 'settings';
    src:
        url(data:application/x-font-ttf;charset=utf-8;base64,AAEAAAAKAIAAAwAgTlMvMj0gSRsAAAEoAAAAVmNtYXDNEdVAAABiAAAADZnbHlm7/n
        pHAAAAAcgAAAEwaGVhZBKtDIMAAADQAAAAANmhoZWEHmQNRAAAArAAAACRobXR4B+gAAAAAYAAAAA
        IbG9jYQCYAAAAAHAHAABmlheHABDgB0AAABCAAAACBuYW11xmFdywAAAvgAAAIxcG9zdDwSCic
        AAAUsAAAAANwABAAADUv9qAFoEAAAA//4D6gABAAAAAAAAAAAAAAAAAAgABAAAAAQAAMLNyX18
        PPPUACwPoAAAAANfSY9oAAAAA19Jj2gAAAAAD6gPqAAAAACAACAAAAAAAAAAAAACAGgAAgAAAAA
        AAQAAAAoACgAAAP8AAAAAAAAAAAAOP0AZAABQAAAAAnoCvAAAAIwCeqK8AAAB4AAxAOIAAAIABOMAAAAA
```

```
AAAAAAAAAAAAAAAAAUGFZABA5wDnAANS/2oAWgPqAJYAAAAABAAAAAAAPoAAA  
AAAAACAAAAAwAAABQAAwABAAAAFAAEACTIAAAEEAAQAQA5wD//wAA5wD//wAAAAEABAAAAAEAAAA  
AAAAAmAAAAAIAAAAAA+oD6gALAGcAAAEOAQcuASc+ATceAQEVBgcnJiIPAQYUHWEQAQcjIgYdAR4  
BOWEFWcGFB8BFjI/AR4BFxUeAtSBPgE9ATY3FxYyPwE2NC8BPGE3MzI2PQE0JisBJic3NjQvASY  
idWeuAsclLgEnIw4BAR4CcVVUCQMDCVRVcf7pOTRbBRQGUQYGWhAYB30LDgEPCX0OIVoHB1EFFQV  
bGTyeAQ4KcQsOOTRbBRQGUQYGWhAXB34LDQ8Jfg4gWgcCHUQUVBVsZNh4BDgpxCQ8B9lRxAwNxVFV  
xAgJxAYd+DiBaBgZRBRUFWBk2Hg8KcQsOOTRbBxQHUYQGWhAYB30LDgEPCX0OIVoGBk4FFQVbGTy  
eDwpxCw06NFoIEwhRBgzAEBghfQsNAQENAAAAABIA3gABAAAAAAAAAAAAEAAAABAAAAAAABAAgAAQA  
BAAAAAAAAAACAACACQABAAAAAADAAgAEAABAAAAAAAEAAgAGAABAAAAAAAFAAAsAIAABAAAAAAAGAAg  
AKwABAAAAAAAKAcwAMwABAAAAAALABIAXwADAAEEECQAAAAIacQADAEEECQABABAAcwADAAEEECQA  
CAA4AgwADAAEEECQADABAAkQADAEEECQAEABAAoQADAEEECQAFABYAsQADAEEECQAGABAAxwADAAE  
ECQAKAfGa1wADAAEEECQALACQLyBzZXROaW5nc1JlZ3VsYXJzZXROaW5nc3NdHRpbmdzMmVyc2l  
vb2luZS1kaG9zaGVhZGVzZXROaW5nc0Zvb2luZS1kaG9zaGVhZGVzZXROaW5nc0Zvb2luZS1kaG9  
vd3d3LnN5bmNmZDhXNpb24uY29tACAAcwBlAHQAdABpAG4AZwBzAFIAZQBnAHUAAbABhAHIAcwBlAHQ  
AdABpAG4AZwBzAHMAZQB0AHQAaQBuAGcAcwBWAGUAcgBzAGkAbwBuACAAMQAuADAAcwBlAHQAdAB  
pAG4AZwBzAEYAwbwBuAHQAIAbnAGUAbGABgBlAHIAAYQBOAGUAZAAGAHUAcwBpAG4AZwAgAFMAEQBuAGM  
AZgBlAHMAaQBVAG4AIAbnAGUAdABYAG8AIAbtAHQAdQBkAGkAbwB3AHcAdwAuAHMAEQBuAGMAZgB  
lAHMAaQBVAG4ALGbjAG8AbQAAAAAACAAAAAAAOAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAIBAgE  
DAAlzZXROaW5ncy0tLTExAAAA) format('trueType');  
  
font-weight: normal;  
font-style: normal;  
}  
.e-btn-icons {  
font-family: 'settings' !important;  
speak: none;  
font-size: 55px;  
font-style: normal;  
font-weight: normal;  
font-variant: normal;  
text-transform: none;  
line-height: 1;  
-webkit-font-smoothing: antialiased;  
-moz-osx-font-smoothing: grayscale;  
}  
.e-setting-icon::before {  
content: "\e700";  
}
```

Tooltip for button in ##Platform Name## Button control

Tooltip can be shown on Button hover and it can be achieved by setting title attribute.

The following snippets illustrates how to show tooltip on Button hover.

INDEX.JS

```
ej.base.enableRipple(true);
//To change the Button type.
var button = new ej.buttons.Button({ isPrimary: true, content: "Button" });
button.appendTo('#element');
button.element.setAttribute("title", "Primary Button");
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Button</title>
  <meta charset="utf-8">
```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="element"></button>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}

```

Calendar

Date range in ##Platform_Name## Calendar control

Calendar provides an option to select a date value within a specified range by defining the [min](#) and [max](#) properties. The min date should always be lesser than the max date. If the value of [min](#) or [max](#) properties are changed through code behind, then update the [value](#) property to be set within the specified range, or else, if the value is out of specified date range and less than [min](#) date, value property will be updated with min date or the value is higher than max date, value property will be updated with [max](#) date.

The following example allows you to select a date within the range of 7th to 27th days in a month.

INDEX.JS

```
ej.base.enableRipple(true);

//Creates a Calendar with min and max properties.
var today = new Date();
var currentYear = today.getFullYear();
var currentMonth = today.getMonth();
var currentDay = today.getDate();
var calendar = new ej.calendars.Calendar({
  //sets the min date
  min: new Date(currentYear, currentMonth, 7),
  //sets the max date
  max: new Date(currentYear, currentMonth, 27),
  //sets the value
  value: new Date(new Date().setDate(14))
});
calendar.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Calendar control</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <!--style reference from the Calendar component-->
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-calendars/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <!--element which is going to render the Calendar-->
    <div id="element"></div>
  </div>
  <script>
  var ele = document.getElementById('container');
  if(ele) {
    ele.style.visibility = "visible";
  }
  </script>
```

```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Multi select in ##Platform_Name## Calendar control

Calendar provides an option to select **single** or **multiple dates** by using `isMultiSelection` and `values` properties. By default, `isMultiSelection` property will be in disabled state.

| API | Type | Description |

|-----|-----|-----|

| `isMultiSelection` | **Boolean** | Enables the multi-selection option in the Calendar control |

| `values` | **Date[]** | Gets or sets the date range values in multi-selection option |

The following example demonstrates the functionality of `isMultiSelection` property and `values` properties in the Calendar control.

INDEX.JS

```
/*Initialize the calendar component*/
var calendar = new ej.calendars.Calendar({
    isMultiSelection: true,
    values: [new Date('1/1/2020'), new Date('1/15/2020'), new
Date('1/3/2020'), new Date('1/25/2020')]
});
calendar.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Calendar control</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <!--style reference from the Calendar component-->
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <!--element which is going to render the Calendar-->
```

```

        <div id="element"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Select a sequence of dates in Calendar](#)

Globalization in ##Platform_Name## Calendar control

Globalization is the combination of adapting the component to various languages by means of parsing and formatting the date or number [Internationalization](#) and also by adding cultural specific customizations and translating the text [localization](#)

By default, the Calendar date format, week, and month names are specific to American English culture. It uses the [Essential JavaScript 2 Internationalization](#) package to parse and format date object based on the culture using the official [UNICODE CLDR](#) JSON data. It provides the [loadCldr](#) method to load the culture-specific CLDR JSON data.

All the Essential JS 2 component are specific to English culture ('en-US'). If you want to go with the different culture other than English, follow the below steps.

- Install the `CLDR-Data` package by using the below command (installs the CLDR JSON data). To know more about CLDR data, refer to the [CLDR-Data](#) link.

```
npm install cldr-data --save
```

Once the package is installed, the culture-specific JSON data will be available in `/node_modules/cldr-data`.

- Now, import the installed CLDR JSON data to the `app.ts` file. To import JSON data, install the JSON plugin loader. In the example, SystemJS JSON plugin loader is used.

```
npm install systemjs-plugin-json --save-dev
```

- After it is installed, configure the `system.config.js` settings as given below to map the `systemjs-plugin-json` loader.

```

`ts
System.config({
  paths: {
    'npm:': './node_modules/',
    'syncfusion:': 'npm:@syncfusion/'
  },
  map: {
    app: 'app',
    //Syncfusion packages mapping
    "@syncfusion/ej2-base": "syncfusion:ej2-base/dist/ej2-base.umd.min.js",
    "@syncfusion/ej2-data": "syncfusion:ej2-data/dist/ej2-data.umd.min.js",
    "@syncfusion/ej2-inputs": "syncfusion:ej2-inputs/dist/ej2-inputs.umd.min.js",
    "@syncfusion/ej2-buttons": "syncfusion:ej2-buttons/dist/ej2-buttons.umd.min.js",
    "@syncfusion/ej2-splitbuttons": "syncfusion:ej2-splitbuttons/dist/ej2-splitbuttons.umd.min.js",
    "@syncfusion/ej2-lists": "syncfusion:ej2-lists/dist/ej2-lists.umd.min.js",
    "@syncfusion/ej2-popups": "syncfusion:ej2-popups/dist/ej2-popups.umd.min.js",
    "@syncfusion/ej2-calendars": "syncfusion:ej2-calendars/dist/ej2-calendars.umd.min.js",
    "cldr-data": 'npm:cldr-data',
    //systemjs-plugin-json loader package mapping
    "plugin-json": "npm:systemjs-plugin-json/json.js"
  },
  meta: {
    '*.json': { loader: 'plugin-json' }
  },
  packages: {
    'app': { main: 'app', defaultExtension: 'js' },
    'cldr-data': { main: 'index.js', defaultExtension: 'js' }
  }
});
System.import('app');
`

```

- Now, use the [loadCldr](#) method to load the culture-specific CLDR JSON data from the installed location to `app.ts` file.

- Calendar displayed **Sunday** as the first day of week based on default culture ("en-US"). If you want to display the Calendar with loaded culture's first day of week, you need to import **weekdata.json** file from the **cldr-data/supplemental** as given in the code example.

```
`ts
```

```
//import the loadCldr from ej2-base
```

```
import { loadCldr } from '@syncfusion/ej2-base';
```

```
declare var require: any;
```

```
loadCldr(
```

```
require('cldr-data/supplemental/numberingSystems.json'),
```

```
require('cldr-data/main/de/ca-gregorian.json'),
```

```
require('cldr-data/main/de/numbers.json'),
```

```
require('cldr-data/main/de/timeZoneNames.json'),
```

```
require('cldr-data/supplemental/weekdata.json')); // To load the culture based first day of week
```

```
,
```

The [Localization](#) library allows you to localize default text content of the Calendar. The Calendar component has static text for **today** feature that can be changed to other cultures (Arabic, Deutsch, French, etc.) by defining the [locale](#) value and translation object.

Locale keywords |Text

today | Name of the button to choose Today date.

- Before changing the culture other than **English**, ensure that locale text for the concerned culture is loaded through **load** method of

[L10n](#) class.

```
`ts
```

```
//Load the L10n from ej2-base
```

```
import { L10n } from '@syncfusion/ej2-base';
```

```
//load the locale object to set the localized placeholder value
```

```
L10n.load({
```

```
'de': {
```

```
'calendar': { today: 'heute'}
```

```
}
```

```
});
```

```
,
```

- Set the culture by using the [locale](#) property. The code example initializes the Calendar component in the **German** culture.

```
`ts
import { Calendar } from '@syncfusion/ej2-calendars';
//import the loadCldr from ej2-base
import { loadCldr, L10n } from '@syncfusion/ej2-base';
declare var require: any;
loadCldr(
  require('cldr-data/supplemental/numberingSystems.json'),
  require('cldr-data/main/de/ca-gregorian.json'),
  require('cldr-data/main/de/numbers.json'));
require('cldr-data/main/de/timeZoneNames.json'));
L10n.load({
  'de': {
    'calendar': { today: 'heute' }
  }
});
let calendarObject: Calendar = new Calendar({
  //sets the locale.
  locale: 'de'
});
calendarObject.appendTo('#element');
```

The following example displays the Calendar in **German** culture.

INDEX.JS

```
ej.base.enableRipple(true);
var L10n = ej.base.L10n;
  L10n.load({
    'de': {
      'calendar': { today: 'heute' }
    }
  });
  loadCultureFiles('de');
  var calendar = new ej.calendars.Calendar({
    locale: 'de'
  });
  calendar.appendTo('#element');
  function loadCultureFiles(name) {
```

```

    var files = ['ca-gregorian.json', 'numbers.json',
'timeZoneNames.json'];
    var loader = ej.base.loadCldr;
    var loadCulture = function (prop) {
        var val, ajax;
        ajax = new
ej.base.Ajax('https://ej2.syncfusion.com/javascript/demos/' +
'src/common/cldr-data/main/' + name + '/' + files[prop], 'GET', false);
        ajax.onSuccess = function (value) {
            val = value;
        };
        ajax.send();
        loader(JSON.parse(val));
    };
    for (var prop = 0; prop < files.length; prop++) {
        loadCulture(prop);
    }
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Calendar control</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <!--style reference from the Calendar component-->
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <!--style reference from app-->
    <link href="styles.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <!--element which is going to render the Calendar-->
        <div id="element"></div>
    </div>
    <script>
    var ele = document.getElementById('container');
    if(ele) {
        ele.style.visibility = "visible";
    }

    </script>
    <script src="index.js" type="text/javascript"></script>

```

```
</body></html>
```

Right-to-left

The Calendar supports right-to-left functionality for languages like Arabic, Hebrew, etc. to display text in the right-to-left direction. Use

[enableRtl](#) property to set the RTL direction.

The following code example initializes the Calendar component in **Arabic** culture.

```
`ts
import { Calendar } from '@syncfusion/ej2-calendars';
//import the loadCldr from ej2-base
import { loadCldr, L10n } from '@syncfusion/ej2-base';
declare var require: any;
loadCldr(
  require('cldr-data/supplemental/numberingSystems.json'),
  require('cldr-data/main/ar/ca-gregorian.json'),
  require('cldr-data/main/ar/numbers.json'));
require('cldr-data/main/ar/timeZoneNames.json'));
L10n.load({
  'ar': {
    'calendar': {
      today: 'اليوم'
    }
  }
});
let calendarObject: Calendar = new Calendar({
  //sets the locale.
  locale: 'ar'
});
calendarObject.appendTo('#element');
```

The following example displays the Calendar in **Arabic** culture in the right-to-left direction.

INDEX.JS

```
ej.base.enableRipple(true);
```



```

var L10n = ej.base.L10n;
L10n.load({
  'ar': {
    'calendar': { today: 'اليوم' }
  }
});
loadCultureFiles('ar');
var calendar = new ej.calendars.Calendar({
  locale: 'ar', enableRtl: true
});
calendar.appendTo('#element');

function loadCultureFiles(name) {
  var files = ['ca-gregorian.json', 'numbers.json',
'timeZoneNames.json'];
  if (name === 'ar') {
    files.push('numberingSystems.json');
  }
  var loader = ej.base.loadCldr;
  var loadCulture = function (prop) {
    var val, ajax;
    if (name === 'ar' && prop === files.length - 1) {
      ajax = new
ej.base.Ajax('https://ej2.syncfusion.com/javascript/demos/src/common/cldr-
data/supplemental/' + files[prop], 'GET', false);
    } else {
      ajax = new
ej.base.Ajax('https://ej2.syncfusion.com/javascript/demos/src/common/cldr-
data/main/' + name + '/' + files[prop], 'GET', false);
    }
    ajax.onSuccess = function (value) {
      val = value;
    };
    ajax.send();
    loader(JSON.parse(val));
  };
  for (var prop = 0; prop < files.length; prop++) {
    loadCulture(prop);
  }
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Calendar control</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <!--style reference from the Calendar component-->
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

```

```

<link href="styles.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <!--element which is going to render the Calendar-->
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customization in ##Platform_Name## Calendar control

Each day cell of the Calendar can be customized by using the [renderDayCell](#)

event.

The following section demonstrates how to disable or highlight specific dates in a Calendar.

Disable weekends

You can disable weekends of every month in a Calendar by using the [renderDayCell](#) event. The [renderDayCell](#) event offers the following arguments on each day cell creation to help you disable the dates.

| **View** | **Description** |

| --- | --- |

| **date** | Defines the current date of the Calendar. |

| **isDisabled** | Specifies whether the current date is to be disabled or not. |

| **isOutOfRange** | Defines whether the current date is out of range or not. |

The following example demonstrates how to disable weekends of every month.

INDEX.JS

```

ej.base.enableRipple(true);

var calendar = new ej.calendars.Calendar({
    renderDayCell: disabledDate,
    value: new Date()
});
function disabledDate(args) {
    if (args.date.getDay() === 0 || args.date.getDay() === 6) {

```

```

        //Set 'true' to disable the weekends.
        args.isDisabled = true;
    }
}

calendar.appendTo('#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Calendar control</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <!--style reference from the Calendar component-->
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <!--element which is going to render the Calendar-->
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Day cell format

You can also highlight specific dates by adding custom CSS or element to the day cell by using the [renderDayCell](#) event.

You can customize the appearance of the Calendar by overriding the existing styles. The following list of CSS class names are used to customize the Calendar component.

Class Name	Description
------------	-------------

---	---
-----	-----

- | e-calendar | Applied to the Calendar. |
- | e-header | Applied to the header. |
- | e-title | Applied to the title. |
- | e-icon-container | Applied to the previous and next icon container. |
- | e-prev | Applied to the previous icon. |
- | e-next | Applied to the next icon. |
- | e-weekend | Applied to weekends. |
- | e-other-month | Applied to days of other months. |
- | e-day | Applied to each day cell. |
- | e-selected | Applied to the selected dates. |
- | e-disabled | Applied to the disabled dates. |

The following example highlights the World Health Day (every 7th April) and World Forest Day (every 21st March) by using the custom icon and ToolTip.

INDEX.JS

```
ej.base.enableRipple(true);

var calendar = new ej.calendars.Calendar({
  renderDayCell: customDates,
  value: new Date('03/10/2017')
});
function customDates(args) {
  //Defines the custom HTML element to be added.
  var span = document.createElement('span');
  //Uses "e-icons" class name to load Essential JS 2 built-in icons.
  span.setAttribute('class', 'e-icons highlight-day');
  if (+args.date.getDate() === 7 && +args.date.getMonth() == 3) {
    //Appends the span element to day cell.
    args.element.appendChild(span);
    //Sets the custom tooltip to the special dates.
    args.element.setAttribute('title', 'World health day!');
    //Uses "special" class name to highlight special dates that you can
    refer in "styles.css".
    args.element.className = 'special';
  }
  if (+args.date.getDate() === 21 && +args.date.getMonth() == 2) {
    args.element.appendChild(span);
    args.element.className = 'special';
    //Sets the custom tooltip to the special dates.
    args.element.setAttribute('title', 'World forest day');
  }
}
calendar.appendTo('#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Calendar control</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <!--style reference from the Calendar component-->
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <!--element which is going to render the Calendar-->
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Highlight weekends

You can highlight the weekends of every month in a Calendar by using the [renderDayCell](#) event. The following example demonstrates how to highlights the weekends of every month.

INDEX.JS

```

ej.base.enableRipple(true);

var calendar = new ej.calendars.Calendar({
  renderDayCell: highlightDate,
  value: new Date()
});
function highlightDate(args) {
  if (args.date.getDay() === 0 || args.date.getDay() === 6) {
    //set 'true' to disable the weekends
    args.element.classList.add('e-highlightweekend');
  }
}
calendar.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Calendar control</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <!--style reference from the Calendar component-->
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <!--element which is going to render the Calendar-->
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Add the external button in the Calendar popup](#)
- [How to skip a month in Calendar](#)
- [How to change the first day of week](#)
- [How to customize the calendar day header](#)

Calendar views in ##Platform_Name## Calendar control

The Calendar has the following predefined views that provide a flexible way to navigate back and forth when selecting dates.

| View | Description |

| --- | --- |

| month (default) | Displays the days in a month. |

| year | Displays the months in a year. |

| decade | Displays the years in a decade. |

When view is defined to the [start](#) property of the Calendar, it allows you to set the initial view on rendering.

The following example demonstrates how to set the **year** as the start view of the Calendar.

INDEX.JS

```
ej.base.enableRipple(true);

var calendar = new ej.calendars.Calendar({
    //sets the start view.
    start: "Year",
    //sets the value.
    value: new Date()
});
calendar.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Calendar control</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <!--style reference from the Calendar component-->
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-calendars/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <!--element which is going to render the Calendar-->
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
```

```

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

View restriction

Calendar view navigation can be restricted by defining the [start](#) and [depth](#) property that allows you to select the date from that view.

By defining the start and depth property with the different view, drill-down and drill-up views navigation can be limited to the user. Calendar views will be drill-down up to the view which is set in **depth** property and drill-up up to the view which is set in **start** property.

The following example displays the Calendar in **decade** view, and allows you to select a date in **month** view.

Depth view should always be smaller than the start view. If the views are the same, then the Calendar view remains unchanged.

INDEX.JS

```

ej.base.enableRipple(true);

var calendar = new ej.calendars.Calendar({
    //sets the start view.
    start: "Decade",
    //sets the depth view.
    depth: "Year"
});
calendar.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Calendar control</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <!--style reference from the Calendar component-->
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-calendars/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

```



```

    <div id="container">
      <!--element which is going to render the Calendar-->
      <div id="element"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Accessibility in ##Platform_Name## Calendar control

The Calendar component followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Calendar component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2 Support](#) | |

| [Section 508 Support](#) | |

| [Screen Reader Support](#) | |

| [Right-To-Left Support](#) | |

| [Color Contrast](#) | |

| [Mobile Device Support](#) | |

| [Keyboard Navigation Support](#) | |

| [Accessibility Checker Validation](#) | |

| [Axe-core Accessibility Validation](#) | |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

```

}
</style>

<div> - All
features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The
component does not meet the requirement.</div>

```

WAI-ARIA attributes

The web accessibility makes web content and web applications more accessible for disabled people. It especially helps in dynamic content change and development of advanced user interface controls with AJAX, HTML, JavaScript, and related technologies.

Calendar provides built-in compliance with [WAI-ARIA](#) specifications. WAI-ARIA support is achieved through attributes like `aria-label`, `aria-selected`, `aria-disabled`, and `aria-activedescendant` applied for navigation buttons, disable and active day cells.

It helps disabled persons by providing the information about the widget for assistive technology in the screen readers. Calendar component contains grid role and grid cell for each day cell.

- **Aria-label:** This attribute provides text labels for an object for the previous and next month's elements. It helps the screen reader object to read.
- **Aria-selected:** Indicates the currently selected date of the Calendar component.
- **Aria-disabled:** Indicates the disabled state of the Calendar component.
- **Aria-activedescendant:** Helps in managing the current active child of the Calendar component.
- **Role:** Gives information to assistive technologies about how to handle each element in a widget.
- **Grid-cell:** Defines the individual cell that can be focused and selected.

Keyboard interaction

You can use the following keys to interact with the Calendar. This component implements keyboard navigation support by following the [WAI-ARIA practices](#).

It supports the following list of shortcut keys:

| Press | To do this |

| --- | --- |

| Upper Arrow | Focuses the same day of the previous week. |

| Down Arrow | Focuses the same day of the next week. |

| Left Arrow | Focuses the day before. |

| Right Arrow | Focuses the next day. |

| Home | Focuses the first day of the month. |

| End | Focuses the last day of the month. |

| Page Up | Focuses the same date of the previous month. |

- | Page Down | Focuses the same date of the next month. |
- | Enter | Selects the currently focused date. |
- | Shift + Page Up | Focuses the same date for the previous year. |
- | Shift + Page Down | Focuses the same date for the next year. |
- | Control + Upper Arrow | Moves to the inner level of view like month to year and year to decade. |
- | Control + Down Arrow | Moves out from the depth level view like decade to year and year to month. |
- | Control + Home | Focuses the first date of the current year. |
- | Control + End | Focuses the last date of the current year. |

To focus the Calendar component, use `alt+t` keys.

INDEX.JS

```
ej.base.enableRipple(true);

var calendar = new ej.calendars.Calendar({
});
document.onkeyup = function (e) {
  if (e.altKey && e.keyCode === 84) {
    // press alt+t to focus the control.
    calendar.element.querySelectorAll('.e-content table')[0].focus()
  }
}
calendar.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Calendar control</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <!--style reference from the Calendar component-->
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-calendars/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
```

```

</head>
<body>

    <div id="container">
        <!--element which is going to render the Calendar-->
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Ensuring accessibility

The Calendar component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Calendar component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Calendar component with accessibility tools.

See also

- [Accessibility in Syncfusion components](#)

Islamic calendar in ##Platform_Name## Calendar control

In addition to the Gregorian calendar, the calendar control supports displaying the Islamic calendar (Hijri calendar). **Islamic calendar** or **Hijri calendar** is a **lunar calendar** consisting of 12 months in a year of 354 or 355 days. To know more about Islamic calendar, please refer this [wikipedia](#).

Also, it consists of all Gregorian calendar functionalities as like min and max date, week number, start day of the week, multi selection, enable RTL, start and depth view, localization, highlight and customize the specific dates.

By default, calendar mode is in **Gregorian**. You can enable the Islamic mode by setting the **calendarMode** as **Islamic**. Also, need to import and injecting the **Islamic** module from **ej2-calendars** as shown below.

```

import { Islamic, Calendar } from '@syncfusion/ej2-calendars';\
Calendar.Inject(Islamic);

```

The following example demonstrates how to display the Islamic Calendar (Hijri Calendar).

INDEX.JS

```

ej.base.enableRipple(true);
var span;
var addClass = ej.base.addClass;
var calendar = new ej.calendars.Calendar({
    calendarMode: 'Islamic',
    renderDayCell: disableDate
});

```

```

calendar.appendTo('#element');
function disableDate(args) {
    /*Date need to be disabled*/
    if (args.date.getDate() === 2 || args.date.getDate() === 10 ||
args.date.getDate() === 28) {
        args.isDisabled = true;
    }
    if (args.date.getDate() === 13) {
        span = document.createElement('span');
        args.element.children[0].className += 'e-day sf-icon-cup
highlight';
        addClass([args.element], ['special', 'e-day', 'dinner']);
        args.element.setAttribute('data-val', ' Dinner !');
        args.element.appendChild(span);
    }
    if (args.date.getDate() === 23) {
        args.element.children[0].className += 'e-day sf-icon-start
highlight';
        span = document.createElement('span');
        span.setAttribute('class', 'sf-icons-star highlight');
        //use the imported method to add the multiple classes to the
given element
        addClass([args.element], ['special', 'e-day', 'holiday']);
        args.element.setAttribute('data-val', ' Holiday !');
        args.element.appendChild(span);
    }
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Calendar control</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <!--style reference from the Calendar component-->
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <!--style reference from app-->
    <link href="styles.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <!--element which is going to render the Calendar-->

```

```

        <div id="element"></div>
    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Style appearance in ##Platform_Name## Calendar control

The following content provides the exact CSS structure that can be used to modify the control's appearance based on the user preference.

Customizing the background color for the Calendar

Use the following CSS to customize the background color and border for the Calendar.

```

,

/ To specify background color and border /

```

```

.e-calendar {
background-color: peachpuff;
border: 3px solid red;
}
,

```

Customizing the Calendar date elements on hovering

Use the following CSS to customize the date elements on hovering in the Calendar.

```

,

/ To specify background color, color, and border /

.e-calendar .e-content td:hover span.e-day, .e-calendar .e-content td:focus span.e-day, .e-bigger.e-small
.e-calendar .e-content td:hover span.e-day, .e-bigger.e-small .e-calendar .e-content td:focus span.e-day
{

```

```

background-color: red;
border: 2px solid;
color: #212529;
}
,

```

Customizing the border of date cell grid

Use the following CSS to add the border to the date cell grid.

```

,

/ To specify border /

```

```
.e-calendar .e-content span.e-day, .e-bigger.e-small .e-calendar .e-content span.e-day {  
border: 1px solid;  
}  
,
```

Customizing the Calendar title

Use the following CSS to customize the Calendar title.

```
,  
  
/ To specify color and font size /  
.e-calendar .e-header .e-title, .e-bigger.e-small .e-calendar .e-header .e-title {  
color: black;  
font-size: 20px;  
}  
,
```

Customizing the previous and next icon

Use the following CSS to customize the previous and next icon.

```
,  
  
/ To specify color and border /  
.e-calendar .e-header span, .e-bigger.e-small .e-calendar .e-header span {  
border: 1px solid;  
color: chocolate;  
}  
,
```

Customizing the footer button

Use the following CSS to customize the footer button.

```
,  
  
/ To specify background color, color, and border-color /  
.e-calendar .e-btn.e-today.e-flat.e-primary, .e-calendar .e-css.e-btn.e-today.e-flat.e-primary {  
background-color: red;  
border-color: black;  
color: black;  
}  
,
```

Customizing the selected date cell grid

Use the following CSS to customize the selected date cell grid in Calendar.

/ To specify background color and color /

```
.e-calendar .e-content td.e-focused-date.e-today span.e-day {
background-color: maroon;
color: #fff;
}
```

Customizing the content header in Calendar

Use the following CSS to customize the content header in Calendar.

/ To specify background /

```
.e-calendar .e-content thead, .e-bigger.e-small .e-calendar .e-content thead {
background: aquamarine;
}
```

How To

Set clear button in calendar in `##Platform_Name##` Calendar control

To configure `clear` button in Calendar UI, do the following:

1. To the `created` event of the Calendar, add the required elements to make clear button visible. In the following example, Essential JS 2 button component within `div` element is used.
2. When the `e-footer` class is used, the div tag acts as the footer.
3. Using these button, selected date can be cleared.
4. Bind the required event handler to the button tags to update the value.

Code example is as follows:

INDEX.JS

```
ej.base.enableRipple(true);
var calendar = new ej.calendars.Calendar({
  created: onCreate
});
function onCreate() {
  //creates the custom element for today button.
  var clearBtn = document.createElement('button');
  var footerElement = document.getElementsByClassName('e-footer-container')[0];
  clearBtn.className = 'e-btn e-clear e-flat';
  clearBtn.textContent = 'Clear';
  footerElement.prepend(clearBtn);
  this.element.appendChild(footerElement);
  // custom click handler to update the value property with null values.
```



```

    document.querySelector('.e-footer-container .e-
clear').addEventListener('click', function () {
    calendar.value = null;
    calendar.dataBind();
});
}
calendar.appendTo('#element');
document.getElementById('loader').style.display = "none";
document.getElementById('container').style.visibility = "visible";

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Calendar control</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <!--style reference from app-->
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <!--style reference from the Calendar component-->
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <!--Element which is going to render-->
        <div id="element"></div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

Show dates of other months in ##Platform_Name## Calendar control

The following example demonstrates how to show dates of other months.

Using the styles below, you can bring the dates of other months to visibility from its hidden state.

,

```
.e-calendar .e-content tr.e-month-hide,
.e-calendar .e-content td.e-other-month>span.e-day {
display: block;
}

.e-calendar .e-content td.e-month-hide,
.e-calendar .e-content td.e-other-month {
pointer-events: auto;
touch-action: auto;
}
`
```

INDEX.JS

```
ej.base.enableRipple(true);

var calendar = new ej.calendars.Calendar({
});
calendar.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Calendar control</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <!--style reference from the Calendar component-->
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <!--element which is going to render the Calendar-->
    <div id="element"></div>
  </div>
  <style>
```

```

        .e-calendar .e-content tr.e-month-hide,
        .e-calendar .e-content td.e-other-month>span.e-day {
            display: block;
        }
        .e-calendar .e-content td.e-month-hide,
        .e-calendar .e-content td.e-other-month {
            pointer-events: auto;
            touch-action: auto;
        }
    </style>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Select a sequence of dates in calendar in ##Platform_Name## Calendar control

The following example demonstrates how to select the week dates of chosen date in the Calendar using [values](#) property, when [multiSelection](#) property is enabled. Methods of Moment.js is used in this sample for calculating the start and end of week from the selected date.

INDEX.JS

```

/*Initialize the calendar component*/
var calendar = new ej.calendars.Calendar({
    isMultiSelection: true,
    change: onChange
});
calendar.appendTo('#calendar');
/*selected current week dates when click the button*/
document.getElementById('workweek').addEventListener('click', function () {
    if (calendar.element.classList.contains('week')) {
        calendar.element.classList.remove('week');
    }
    calendar.element.classList.add('workweek');
});
/*selected current week dates when click the button*/
document.getElementById('week').addEventListener('click', function () {
    if (calendar.element.classList.contains('workweek')) {
        calendar.element.classList.remove('workweek');
    }
    calendar.element.classList.add('week');
});
function onChange(args) {
    var startOfWeek = moment(calendar.value).startOf('week');
    var endOfWeek = moment(calendar.value).endOf('week');
    if (calendar.element.classList.contains('workweek')) {
        getWeekArray(startOfWeek.day(1), endOfWeek.day(5));
    }
    else if (calendar.element.classList.contains("week")) {
        getWeekArray(startOfWeek, endOfWeek);
    }
}

```

```
function getWeekArray(startOfWeek, endOfWeek) {
    var days = [];
    var day = startOfWeek;
    while (day <= endOfWeek) {
        days.push(day.toDate());
        day = day.clone().add(1, 'd');
    }
    calendar.values = days;
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Calendar control</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <!--style reference from app-->
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <!--style reference from the Calendar component-->
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <!--Element which is going to render-->
        <div class="content-wrapper" style="height:357px ;display: inline-
block;">
            <div id="calendar" style="display: inline-block;"></div>
            <div class="btncontainer e-btn-group e-vertical">
                <button class="e-btn" id="week">Week</button>
                <button class="e-btn" id="workweek">Work Week</button>
            </div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
```

```
</body></html>
```

Skip a month in calendar in ##Platform_Name## Calendar control

The following example demonstrates how to skip a month in the Calendar while clicking the previous and next icons. In the example below, the [navigated](#) event is used to skip a month with [navigateTo](#) method.

INDEX.JS

```
ej.base.enableRipple(true);

var calendar = new ej.calendars.Calendar({
  navigated: onNavigate
});
function onNavigate(args) {
  var date;
  if (args.event.currentTarget.classList.contains('e-next')) {
    //Increments the month while clicking the next icon.
    date = new Date(args.date.setMonth(args.date.getMonth() + 1));
  }
  if ((args.event.currentTarget).classList.contains('e-prev')) {
    //Decrements the month while clicking the previous icon.
    date = new Date(args.date.setMonth(args.date.getMonth() - 1));
  }
  if (args.view == 'month') {
    calendarObject.navigateTo('month', date);
  }
}

calendar.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Calendar control</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <!--style reference from the Calendar component-->
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-calendars/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>
```

```

    <div id="container">
      <!--element which is going to render the Calendar-->
      <div id="element"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Render the calendar with week numbers in ##Platform_Name## Calendar control

You can enable **weekNumbers** in the Calendar by using the [weekNumber](#) property.

INDEX.JS

```

ej.base.enableRipple(true);

var calendar = new ej.calendars.Calendar({
  weekNumber: true
});
calendar.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Calendar control</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <!--style reference from the Calendar component-->
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-calendars/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <!--element which is going to render the Calendar-->
    <div id="element"></div>
  </div>

```

```

    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Change the first day of week in ##Platform_Name## Calendar control

The Calendar provides an option to change the first day of the week by using the [firstDayOfWeek](#) property. Generally, the day of the week starts from 0 (Sunday) and ends with 6 (Saturday).

By default, the first day of the week is culture specific.

The following example shows the Calendar with **Tuesday** as the first day of the week.

INDEX.JS

```

ej.base.enableRipple(true);

var calendar = new ej.calendars.Calendar({
    //sets the first day of the week.
    firstDayOfWeek: 2
});
calendar.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Calendar control</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <!--style reference from the Calendar component-->
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-calendars/styles/material.css" rel="stylesheet">
    <!--style reference from app-->
    <link href="styles.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="container">
        <!--element which is going to render the Calendar-->

```

```

        <div id="element"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customize the calendar day header in **##Platform_Name##** Calendar control

You can change the format of the day that to be displayed in header using [dayHeaderFormat](#) property. By default, the format is **Short**.

You can find the possible formats on below.

Name	Description
----- -----	
Short	Sets the short format of day name (like Su) in day header.
Narrow	Sets the single character of day name (like S) in day header.
Abbreviated	Sets the min format of day name (like Sun) in day header.
Wide	Sets the long format of day name (like Sunday) in day header.

INDEX.JS

```

var calendarObject = new ej.calendars.Calendar({
    dayHeaderFormat: "Short"
});
calendarObject.appendTo('#element');
var formatLabel = new ej.dropdowns.DropDownList({
    // set the height of the popup element
    popupHeight: '200px',
    // bind the change event
    change: function(args) {
        calendarObject.dayHeaderFormat = args.value;
    }
});
formatLabel.appendTo('#select');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Calendar control</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <!--style reference from the Calendar component-->
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">

```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <!--element which is going to render the Calendar-->
        <div id="element"></div>
    </div>
    <div id="format">
        <label class="custom-input-label">Header Format Types</label>
        <div id="wrapper">
            <select id="select" class="form-control">
                <option value="Short" selected="">Short</option>
                <option value="Narrow">Narrow</option>
                <option value="Abbreviated">Abbreviated</option>
                <option value="Wide">Wide</option>
            </select>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Card

Header content in ##Platform_Name## Card control

Header

The Card can be created with header title, sub title and images. For adding header you need to create div element with the class `e-card-header` added.

Card provides below elements and corresponding class definitions to include header.

Elements | Description

`e-card-header-caption` | To group the title and subtitle within the header which acts as wrapper.

e-card-header-title | Main title text with in the header.

e-card-sub-title | A sub-title within the header.

e-card-header-image | To include heading image within the header.

e-card-corner | To add rounded corner for the image.

Title and Subtitle

For adding header to the Card , you need to create wrapper **div** element with **e-card-header-caption** class.

- Place the **div** element with **e-card-header-title** class inside the header caption for adding main title.
- Place the **div** element with **e-card-sub-title** class inside the header caption element for adding sub-title.

Image

Card header has an option for adding images in the header. It is aligned with either before or after the header based on the HTML element positioned in the header structure.

- The header image can be added by creating a **div** element with **e-card-header-image** class which can be placed before or after the header caption wrapper element.

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Card Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link rel="shortcut icon" href="resources/favicon.ico">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
  <link href="index.css" rel="stylesheet">
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div style="margin: 50px;">
    <div tabindex="0" class="e-card">
      <div class="e-card-header">
        <div class="e-card-header-image football"></div>
        <div class="e-card-header-caption">
          <div class="e-card-header-title"> Laura Callahan</div>
          <div class="e-card-sub-title">Sales Coordinator and
Representative</div>
        </div>
      </div>
    </div>
  </div>
```

```

</div>
<div style="margin-left: 50px;margin-top:30px">
  <div tabindex="0" class="e-card">
    <div class="e-card-header e-card-corner">
      <div class="e-card-header-caption">
        <div class="e-card-header-title"> Laura Callahan</div>
        <div class="e-card-sub-title">Sales Coordinator and
Representative</div>
      </div>
      <div class="e-card-header-image football"></div>
    </div>
  </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Content

Content in Card holds texts, images, links and all possible HTML elements. Its adaptable within the Card root element.

- Create a `div` element with the class `e-card-content`.
- Place content `div` element in the Card root element or within any Card inner elements.

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Card Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link rel="shortcut icon" href="resources/favicon.ico">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
  <link href="index.css" rel="stylesheet">
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div style="margin: 50px;">
    <!--element which is going to render the Card-->
    <div tabindex="0" class="e-card">
      <div class="e-card-header">
        <div class="e-card-header-image football"></div>
        <div class="e-card-header-caption">

```

```

        <div class="e-card-header-title"> Laura Callahan</div>
        <div class="e-card-sub-title">Sales Coordinator and
Representative</div>
    </div>
</div>
<div class="e-card-content">
    Laura received a BA in psychology from the University of
Washington. She has also completed a course in business French. She reads
and writes French.
</div>
</div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Card image in ##Platform_Name## Card control

Images

The Card supports to include images within the elements, you can add image as direct element anywhere inside card root by adding the `e-card-image` class to `div` element. Using the class defined, you can write CSS styles to load images to that element.

By default, card images occupies full width of its parent element.

,

```

<div class = "e-card">
<div class="e-card-image">
</div>
</div>

```

,

Title

Card image is supported to include a title or caption for the image. By default, Title is placed over the image on left-bottom position with overlay.

,

```

<div class = "e-card">
<div class="e-card-image">
<div class="e-card-title"></div>
</div>
</div>

```

,

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Card Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link rel="shortcut icon" href="resources/favicon.ico">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
  <link href="index.css" rel="stylesheet">
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div style="margin: 50px;">
    <!--element which is going to render the Card-->
    <div class="e-card">
      <div class="e-card-image">
        <div class="e-card-title">JavaScript </div>
      </div>
      <div class="e-card-content"> JavaScript Succinctly was written
to give readers an accurate, concise examination of JavaScript objects and
their supporting nuances, such as complex values, primitive values, scope,
inheritance, the head object, and more. </div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Divider

Divider used to separate the elements inside the card. You can add divider inside the card elements to separate it.

- Place the `div` element with `e-card-separator` class inside the card element for adding a divider.

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Card Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">

```

```

<link rel="shortcut icon" href="resources/favicon.ico">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div style="margin: 50px;">
    <div tabindex="0" class="e-card" id="basic">
      <div class="e-card-title">Explore Cities</div>
      <div class="e-card-separator"></div>
      <div class="e-card-content">
        Sydney is a city on the east coast of Australia. Sydney is
the capital city of New South Wales. About four million people
        live in Sydney which makes it the biggest city in Oceania.
      </div>
      <div class="e-card-separator"></div>
      <div class="e-card-content">
        New York City has been described as the cultural, financial,
and media capital of the world, and exerts a significant impact
        upon commerce and etc.,
      </div>
      <div class="e-card-separator"></div>
      <div class="e-card-content">
        Malaysia is one of the Southeast Asian countries, on a
peninsula of the Asian continent, to a certain extent; it can be recognized
        as part of the Asian continent.
      </div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [How to customize the card image title position](#)

Action buttons in ##Platform_Name## Card control

You can include Action buttons within the Card and customize them. Action button is a `div` element with `e-card-actions` class followed by button tag or anchor tag within the card root element.

- For adding action buttons you can create button or anchor tag with `e-card-btn` class within the card action element.

```

<div class = "e-card">
<div class="e-card-actions">
<button class="e-card-btn"></button>
<a href="#"></a>
</div>
</div>

```

Vertical

By default, action buttons positioned in horizontal alignment , and also it can be aligned to show in vertical alignment by adding `e-card-vertical` class.

```

<div class = "e-card">
<div class="e-card-actions e-card-vertical">
<button class="e-card-btn">More</button>
<a href="#">Share</a>
</div>
</div>

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Card Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link rel="shortcut icon" href="resources/favicon.ico">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div style="margin: 50px;">
    <!--element which is going to render the Card-->
    <div class="e-card" style="max-width:400px">
      <div class="e-card-header-title">Eiffel Tower</div>
      <div class="e-card-content">

```

```

        The Eiffel Tower is acknowledged as the universal symbol of
        Paris and France.
    </div>
    <div class="e-card-actions">
        <button class="e-card-btn">
            
        </button>
        <button class="e-card-btn">
            
        </button>
        <button class="e-card-btn">
            
        </button>
    </div>
</div>
</div>
<div style="margin-left: 50px;">
    <!--element which is going to render the Card-->
    <div class="e-card" style="max-width:400px">
        <div class="e-card-header-title">Eiffel Tower</div>
        <div class="e-card-content">
            The Eiffel Tower is acknowledged as the universal symbol
of Paris and France.
        </div>
        <div class="e-card-actions e-card-vertical">
            <button class="e-card-btn">LIKE</button>
            <button class="e-card-btn">SHARE</button>
        </div>
    </div>
</div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [How to integrate other component inside the card](#)

Horizontal in ##Platform_Name## Card control

By default, all the card elements are aligned vertically one after the other as in the DOM. You can achieve the element to align horizontally as well by adding the class `e-card-horizontal` in the root card element.

Stacked cards

- An horizontally aligned card can push a specific column to align vertical using `e-card-stacked` class. This will align the stacked section vertically aligned differentiating from horizontal layout.

Class | Description

`e-card-horizontal` | To align card elements horizontally.

`e-card-stacked` | To align elements vertically within the horizontal layout.

```
,
<div tabindex="0" class="e-card e-card-horizontal">
 --> Aligned in horizontal
<div class="e-card-stacked">      --> Aligned in horizontal
// Inside the element all are aligned vertical directions
</div>
</div>
,
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Card Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link rel="shortcut icon" href="resources/favicon.ico">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div style="margin: 50px;display: flex;flex-direction: row;justify-
content: center;">
    <!--element which is going to render the Card-->
    <div tabindex="0" class="e-card e-card-horizontal"
style="width:400px">
      
      <div class="e-card-stacked">
        <div class="e-card-header">
          <div class="e-card-header-caption">
            <div class="e-card-header-title">Philips
Trimmer</div>
          </div>
        </div>
      </div>
    </div>
  </div>
```

```

        <div class="e-card-content">
            Powered by the innovative DuraPower Technology which
            optimizes power consumption, Philips trimmers are designed to last longer
            than 4 ordinary trimmers.
        </div>
    </div>
</div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Style in ##Platform_Name## Card control

The following content provides the exact CSS structure that can be used to modify the control's appearance based on user preference.

Customizing the card

Use the following CSS to customize the card properties.

```

,

.e-card {
background-color: aqua;
padding-left: 20px;
margin-bottom: 20px;
}
,

```

Customizing the Header element

Use the following CSS to customize the Header element properties.

```

,

.e-card .e-card-header {
font-family: cursive;
font-style: italic;
}
,

```

Customizing the card content

Use the following CSS to customize the card content properties.

```

,

.e-card .e-card-content {

```

```
font-size: 20px;
color: gray;
line-height: initial;
font-weight: normal;
}
`
```

Divider used to separate the elements inside the card

Use the following CSS to customize the Divider used to separate the elements inside the card properties.

```
`
.e-card .e-card-separator {
padding-bottom: 30px;
}
`
```

Including image within card element

Use the following CSS to Include image within card element.

```
`
.e-card .e-card-image {
background-image: url(images.png);
background-color: yellow;
height: 160px;
}
`
```

Including a title or caption for the image

Use the following CSS to Include a title or caption for the image.

```
`
.e-card .e-card-image .e-card-title {
font-family: cursive;
font-style: italic;
}
`
```

To include heading image within the header

Use the following CSS to Include heading image within the header.

```
`
.e-card .e-card-header .e-card-header-image {
```

```
height: 48px;
```

```
width: 48px;
```

```
}
```

```
,
```

Customizing the Header main title

Use the following CSS to Customize the Header main title.

```
,
```

```
.e-card .e-card-header .e-card-header-caption .e-card-header-title {
```

```
font-size: large;
```

```
color: aquamarine;
```

```
}
```

```
,
```

Customizing the Header subtitle

Use the following CSS to Customize the Header subtitle.

```
,
```

```
.e-card .e-card-header .e-card-header-caption .e-card-sub-title {
```

```
font-size: 20px;
```

```
font-variant: all-petite-caps;
```

```
}
```

```
,
```

Including action buttons or anchor tags

Use the following CSS to Include action buttons or anchor tags.

```
,
```

```
.e-card .e-card-actions .e-card-btn {
```

```
padding-left: 20px;
```

```
background-color: wheat;
```

```
}
```

```
,
```

To align card elements horizontally

Use the following CSS to align card elements horizontally.

```
,
```

```
.e-card .e-card-horizontal {
```

```
margin: auto;
```

```
width: inherit;
```

```
}
,
```

To align elements vertically within the horizontal layout

Use the following CSS to align elements vertically within the horizontal layout.

```
,
```

```
.e-card .e-card-horizontal .e-card-stacked {
```

```
justify-content: flex-start;
```

```
margin: initial;
```

```
}
```

```
,
```

How To

Customize the card image title position in ##Platform_Name## Card control

Card Image titles are placed as always Bottom-Left Corner only, You can manually customize to placing titles anywhere over the image by adding styles.

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>

  <title>Essential JS 2 Card Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link rel="shortcut icon" href="resources/favicon.ico">
  <link href="//cdn.syncfusion.com/ej2/21.2.3/material.css"
rel="stylesheet">
  <link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div>

    <div id="container">
      <!--element which is going to render the Card-->
      <div class="e-card">
        <div class="e-card-image">
          <div class="e-card-title">Node.js</div>
        </div>
        <div class="e-card-content">
          Node.js is a wildly popular platform for writing web
applications that has revolutionized web development in many ways, enjoying
```

```

                                support across the open source community as well as
industry.
                                </div>
                                </div>
                                </div>
                                </div>
                                <div style="Margin: 5px 0;width:300px">
                                <select id="title_position">
                                <option value="bottom-left">BottomLeft</option>
                                <option value="top-left">TopLeft</option>
                                <option value="top-right">TopRight</option>
                                <option value="bottom-right">BottomRight</option>
                                </select>
                                </div>
                                <script>
                                var ele = document.getElementById('container');
                                if(ele) {
                                ele.style.visibility = "visible";
                                }
                                </script>
                                <script src="index.js" type="text/javascript"></script>
                                </body></html>

```

INDEX.JS

```

// initialize DropDownList component
let dropDownListObject = new ej.dropdowns.DropDownList({
    placeholder:"Select Position",
    change: changed,
});
// render initialized DropDownList
dropDownListObject.appendTo('#title_position');
function changed(e) {
    var cardEle = document.querySelector('.e-card');
    var titleEle = cardEle.querySelector('.e-card-image .e-card-title');
    titleEle.className = '';
    titleEle.classList.add('e-card-title');
    titleEle.classList.add('e-card-' + e.value);
}

```

Integrate other component inside the card in ##Platform_Name## Card control

You can integrate any component inside the card element. Here ListView component is placed inside the card for showcasing the To-Do list.

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Card Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Essential JS 2">
    <meta name="author" content="Syncfusion">
    <link rel="shortcut icon" href="resources/favicon.ico">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div style="margin: 50px;">
    <div id="container">
      <div tabindex="0" class="e-card" id="basic">
        <div class="e-card-title">To-Do List</div>
        <div class="e-card-separator"></div>
        <div class="e-card-content">
          <div id="element"></div>
        </div>
      </div>
    </div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Carousel

Populating items in ##Platform_Name## Carousel control

In the Carousel, slides can be rendered in two ways as follows,

- Populating items using carousel item
- Populating items using data source

Populating items using carousel item

When rendering the Carousel component using items binding, you can assign templates for each item separately or assign a common template to each item. You can also customize the slide transition interval for each item separately. The following example code depicts the functionality as item property binding.

INDEX.JS

```

var carouselObj = new ej.navigations.Carousel({
  items: [
    { template: '<figure class="img-container"><figcaption class="img-
caption">Cardinal</figcaption></figure>' },

```

```

    { template: '<figure class="img-container"><figcaption class="img-
caption">Kingfisher</figcaption></figure>' },
    { template: '<figure class="img-container"><figcaption
class="img-caption">Keel-billed-toucan</figcaption></figure>' },
    { template: '<figure class="img-container"><figcaption
class="img-caption">Yellow-warbler</figcaption></figure>' },
    { template: '<figure class="img-container"><figcaption class="img-
caption">Bee-eater</figcaption></figure>' }
  ]
});
carouselObj.appendTo('#carousel');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.cs
s" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="control-section">
      <div class="control-container">
        <div id="carousel"></div>
      </div>
    </div>
  </div>

```



```

    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Populating items using data source

When rendering the Carousel component using data binding, you can assign a common template only for all items using the `itemTemplate` property. You cannot set the interval for each item. The following example code depicts the functionality as data binding.

INDEX.JS

```

window.getImage = function (bird) {
    return
    `https://ej2.syncfusion.com/products/images/carousel/${bird}.png`;
};
var productItems = [
    { ID: 1, Name: "Cardinal", imageName: 'cardinal' },
    { ID: 2, Name: "Kingfisher", imageName: 'hunei' },
    { ID: 3, Name: "Keel-billed-toucan", imageName: 'costa-rica' },
    { ID: 4, Name: "Yellow-warbler", imageName: 'kaohsiung' },
    { ID: 5, Name: "Bee-eater", imageName: 'bee-eater' }
];
var carouselObj = new ej.navigations.Carousel({
    dataSource: productItems,
    itemTemplate: '<div class="fs-5"><figcaption class="img-
caption">${Name}</figcaption></div>'
});
carouselObj.appendTo('#carousel');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Essential JS 2">
    <meta name="author" content="Syncfusion">
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.cs
s" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <!--style reference from app-->

```

```

<link href="styles.css" rel="stylesheet">
<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="control-section">
            <div class="control-container">
                <div id="carousel"></div>
            </div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Selection

The Carousel items will be populated from the first index of the Carousel items and can be customized using the following ways,

- Select an item using the property.
- Select an item using the method.

Select an item using the property

Using the [selectedIndex](#) property of the Carousel component, you can set the slide to be populated at the time of initial rendering else you can switch to the particular slide item.

INDEX.JS

```

var carouselObj = new ej.navigations.Carousel({
    items: [
        { template: '<figure class="img-container"><figcaption class="img-
caption">Cardinal</figcaption></figure>' },
        { template: '<figure class="img-container"><figcaption class="img-
caption">Kingfisher</figcaption></figure>' },
        { template: '<figure class="img-container"><figcaption
class="img-caption">Keel-billed-toucan</figcaption></figure>' },

```

```

    { template: '<figure class="img-container"><figcaption
class="img-caption">Yellow-warbler</figcaption></figure>' },
    { template: '<figure class="img-container"><figcaption class="img-
caption">Bee-eater</figcaption></figure>' }
  ],
  selectedIndex: 3
});
carouselObj.appendTo('#carousel');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.cs
s" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="control-section">
      <div class="control-container">
        <div id="carousel"></div>
      </div>
    </div>
  </div>
</body>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>

```

```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Select an item using the method

Using the [prev](#) or [next](#) public method of the Carousel component, you can switch the current populating slide to a previous or next slide.

INDEX.JS

```
var carouselObj = new ej.navigations.Carousel({
  items: [
    { template: '<figure class="img-container"><figcaption class="img-
caption">Cardinal</figcaption></figure>' },
    { template: '<figure class="img-container"><figcaption class="img-
caption">Kingfisher</figcaption></figure>' },
    { template: '<figure class="img-container"><figcaption
class="img-caption">Keel-billed-toucan</figcaption></figure>' },
    { template: '<figure class="img-container"><figcaption
class="img-caption">Yellow-warbler</figcaption></figure>' },
    { template: '<figure class="img-container"><figcaption class="img-
caption">Bee-eater</figcaption></figure>' }
  ]
});
carouselObj.appendTo('#carousel');
var prevButton = new ej.buttons.Button({ cssClass: "e-info" });
prevButton.appendTo("#prev");
prevButton.element.onclick = function () {
  carouselObj.prev();
};
var nextButton = new ej.buttons.Button({ cssClass: "e-info" });
nextButton.appendTo("#next");
nextButton.element.onclick = function () {
  carouselObj.next();
};
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
```

```

<link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.cs
s" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="control-section">
      <div id="prev">Previous</div>
      <div id="next">Next</div>
      <div class="control-container">
        <div id="carousel"></div>
      </div>
    </div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Partial visible slides

The Carousel component supports to show one complete slide and a partial view of adjacent (previous and next) slides at the same time. You can enable or disable the partial slides using the [partialVisible](#) property.

INDEX.JS

```

var carouselObj = new ej.navigations.Carousel({
  items: [
    { template: '<figure class="img-container"><figcaption class="img-
caption">Cardinal</figcaption></figure>' },
    { template: '<figure class="img-container"><figcaption class="img-
caption">Kingfisher</figcaption></figure>' },

```

```

    { template: '<figure class="img-container"><figcaption
class="img-caption">Keel-billed-toucan</figcaption></figure>' },
    { template: '<figure class="img-container"><figcaption
class="img-caption">Yellow-warbler</figcaption></figure>' },
    { template: '<figure class="img-container"><figcaption class="img-
caption">Bee-eater</figcaption></figure>' }
  ],
  partialVisible: true
});
carouselObj.appendTo('#carousel');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.cs
s" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="control-section">
      <div class="control-container">
        <div id="carousel"></div>
      </div>
    </div>
  </div>
<script>
var ele = document.getElementById('container');

```

```

if(ele) {
    ele.style.visibility = "visible";
}

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Slide animation only applicable if the `partialVisible` is enabled.

The last slide will be displayed as a partial slide at the initial rendering when the `loop` and `partialVisible` properties are enabled.

The previous slide is not displayed at the initial rendering when the `loop` is disabled.

The following example code depicts the functionality of `partialVisible` and without `loop` functionalities.

INDEX.JS

```

var carouselObj = new ej.navigations.Carousel({
    items: [
        { template: '<figure class="img-container"><figcaption class="img-
caption">Cardinal</figcaption></figure>' },
        { template: '<figure class="img-container"><figcaption class="img-
caption">Kingfisher</figcaption></figure>' },
        { template: '<figure class="img-container"><figcaption
class="img-caption">Keel-billed-toucan</figcaption></figure>' },
        { template: '<figure class="img-container"><figcaption
class="img-caption">Yellow-warbler</figcaption></figure>' },
        { template: '<figure class="img-container"><figcaption class="img-
caption">Bee-eater</figcaption></figure>' }
    ],
    partialVisible: true
});
carouselObj.appendTo('#carousel');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Essential JS 2">
    <meta name="author" content="Syncfusion">

```

```

<link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.cs
s" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="control-section">
      <div class="control-container">
        <div id="carousel"></div>
      </div>
    </div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Customizing partial slides size](#)

Navigators and indicators in ##Platform_Name## Carousel control

The navigators and indicators are used to transition the slides manually.

Navigators

Show or hide previous and next button

In navigators, the previous and next slide transition buttons are used to perform slide transitions manually. You can show/hide the navigators using the [buttonsVisibility](#) property. The possible property values are as follows:

- **Hidden** – the navigator's buttons are not visible.
- **Visible** – the navigator's buttons are visible.
- **VisibleOnHover** – the navigator's buttons are visible only when hovering over the carousel.

The following example depicts the code to show/hide the navigators in the carousel.

INDEX.JS

```
var carouselObj = new ej.navigations.Carousel({
  items: [
    { template: '<figure class="img-container"><figcaption class="img-
caption">Cardinal</figcaption></figure>' },
    { template: '<figure class="img-container"><figcaption class="img-
caption">Kingfisher</figcaption></figure>' },
    { template: '<figure class="img-container"><figcaption
class="img-caption">Keel-billed-toucan</figcaption></figure>' },
    { template: '<figure class="img-container"><figcaption
class="img-caption">Yellow-warbler</figcaption></figure>' },
    { template: '<figure class="img-container"><figcaption class="img-
caption">Bee-eater</figcaption></figure>' }
  ],
  buttonsVisibility: 'Visible'
});
carouselObj.appendTo('#carousel');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.cs
s" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="control-section">
            <div class="control-container">
                <div id="carousel"></div>
            </div>
        </div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Show previous and next button on hover

In the carousel, you can show the previous and next buttons only on mouse hover using the [buttonsVisibility](#) property. The following example depicts the code to show the navigators on mouse hover in the carousel.

INDEX.JS

```

var carouselObj = new ej.navigations.Carousel({
    items: [
        { template: '<figure class="img-container"><figcaption class="img-
caption">Cardinal</figcaption></figure>' },
        { template: '<figure class="img-container"><figcaption class="img-
caption">Kingfisher</figcaption></figure>' },
        { template: '<figure class="img-container"><figcaption
class="img-caption">Keel-billed-toucan</figcaption></figure>' },
        { template: '<figure class="img-container"><figcaption
class="img-caption">Yellow-warbler</figcaption></figure>' },
        { template: '<figure class="img-container"><figcaption class="img-
caption">Bee-eater</figcaption></figure>' }
    ],
    buttonsVisibility: 'VisibleOnHover'
});
carouselObj.appendTo('#carousel');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.cs
s" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="control-section">
      <div class="control-container">
        <div id="carousel"></div>
      </div>
    </div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Previous and next button template

Template options are provided to customize the previous button using [previousButtonTemplate](#) and the next button using [nextButtonTemplate](#). The following example depicts the code for applying the template to previous and next buttons in the carousel.

INDEX.JS

```

var carouselObj = new ej.navigations.Carousel({
  items: [

```

```

    { template: '<figure class="img-container"><figcaption class="img-
caption">Cardinal</figcaption></figure>' },
    { template: '<figure class="img-container"><figcaption class="img-
caption">Kingfisher</figcaption></figure>' },
    { template: '<figure class="img-container"><figcaption
class="img-caption">Keel-billed-toucan</figcaption></figure>' },
    { template: '<figure class="img-container"><figcaption
class="img-caption">Yellow-warbler</figcaption></figure>' },
    { template: '<figure class="img-container"><figcaption class="img-
caption">Bee-eater</figcaption></figure>' }
  ],
  previousButtonTemplate: '<button id="previous"></button>',
  nextButtonTemplate: '<button id="next"></button>'
});
carouselObj.appendTo('#carousel');
var prevButton = new ej.buttons.Button({ cssClass: 'e-flat e-round',
iconCss: 'e-icons e-chevron-left-double' });
prevButton.appendTo('#previous');
var nextButton = new ej.buttons.Button({ cssClass: 'e-flat e-round',
iconCss: 'e-icons e-chevron-right-double' });
nextButton.appendTo('#next');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link
href="https://cdn.jsdelivrivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.cs
s" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="control-section">
            <div class="control-container">
                <div id="carousel"></div>
            </div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Indicators

Show or hide indicators

In indicators, the total slides and current slide state have been depicted. You can show/hide the indicators using the [showIndicators](#) property. The following example depicts the code to show/hide the indicators in the carousel.

INDEX.JS

```

var carouselObj = new ej.navigations.Carousel({
    items: [
        { template: '<figure class="img-container"><figcaption class="img-
caption">Cardinal</figcaption></figure>' },
        { template: '<figure class="img-container"><figcaption class="img-
caption">Kingfisher</figcaption></figure>' },
        { template: '<figure class="img-container"><figcaption
class="img-caption">Keel-billed-toucan</figcaption></figure>' },
        { template: '<figure class="img-container"><figcaption
class="img-caption">Yellow-warbler</figcaption></figure>' },
        { template: '<figure class="img-container"><figcaption class="img-
caption">Bee-eater</figcaption></figure>' }
    ],
    showIndicators: true

```

```
});
carouselObj.appendTo('#carousel');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.cs
s" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="control-section">
      <div class="control-container">
        <div id="carousel"></div>
      </div>
    </div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Indicators Template

Template option is provided to customize the indicators by using the [indicatorTemplate](#) property. The following example depicts the code for applying a template to indicators in the carousel.

INDEX.JS

```

var carouselObj = new ej.navigations.Carousel({
  items: [
    { template: '<figure class="img-container"><figcaption class="img-
caption">Cardinal</figcaption></figure>' },
    { template: '<figure class="img-container"><figcaption class="img-
caption">Kingfisher</figcaption></figure>' },
    { template: '<figure class="img-container"><figcaption
class="img-caption">Keel-billed-toucan</figcaption></figure>' },
    { template: '<figure class="img-container"><figcaption
class="img-caption">Yellow-warbler</figcaption></figure>' },
    { template: '<figure class="img-container"><figcaption class="img-
caption">Bee-eater</figcaption></figure>' }
  ],
  indicatorsTemplate: '#indicatorTemplate',
  slideChanged: (args) => {
    var indicators = carouselObj.element.querySelector('.e-carousel-
indicators');
    ej.base.removeClass(indicators.querySelectorAll('.indicator'),
'active');
    ej.base.addClass([(indicators.querySelector('[data-index="' +
args.currentIndex + '"]').children[0])], 'active');
  }
});
carouselObj.appendTo('#carousel');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link
href="https://cdn.jsdelivrivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.cs
s" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

```

```

<script type="text/x-template" id="indicatorTemplate">
  <div class="indicator" indicator-index="{index}"></div>
</script><script
src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="control-section">
      <div id="prev"></div>
      <div id="next"></div>
      <div class="control-container">
        <div id="carousel"></div>
      </div>
    </div>
  </div>
</script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Showing preview of slide in indicator

You can customize the indicators by showing the preview image of each slide using the [indicatorTemplate](#) property. The following example depicts the code for showing the preview image using a template for indicators in the carousel.

INDEX.JS

```

window.getContent = function (index) {
  var slides = ["Slide 1", "Slide 2", "Slide 3", "Slide 4", "Slide 5"];
  return slides[index];
};
var carouselObj = new ej.navigations.Carousel({
  items: [
    { template: '<div class="slide-content">Slide 1</div>' },
    { template: '<div class="slide-content">Slide 2</div>' },
    { template: '<div class="slide-content">Slide 3</div>' },
    { template: '<div class="slide-content">Slide 4</div>' },
    { template: '<div class="slide-content">Slide 5</div>' },
  ],
  indicatorsTemplate: "#indicatorTemplate",
});
carouselObj.appendTo('#carousel');

```

INDEX.HTML


```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.cs
s" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

<script type="text/x-template" id="indicatorTemplate">
  <div class="indicator" indicator-index="{index}">
    <div class="preview-content">${getContent(data.index)}</div>
  </div>
</script><script
src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="control-section">
      <div id="prev"></div>
      <div id="next"></div>
      <div class="control-container">
        <div id="carousel"></div>
      </div>
    </div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Indicators Types

Choose different types of indicators available using the [indicatorsType](#) property. The indicator types are categorized as follows:

- [Default Indicator](#)
- [Dynamic Indicator](#)
- [Fraction Indicator](#)
- [Progress Indicator](#)

Default Indicator

A default indicator in a carousel is a set of dots that indicate the current position of the slide in the carousel. The Default indicator can be achieved by setting the [indicatorsType](#) to **Default**.

INDEX.JS

```
var carouselObj = new ej.navigations.Carousel({
  items: [
    { template: '<figure class="img-container"><figcaption class="img-
caption">Cardinal</figcaption></figure>' },
    { template: '<figure class="img-container"><figcaption class="img-
caption">Kingfisher</figcaption></figure>' },
    { template: '<figure class="img-container"><figcaption
class="img-caption">Keel-billed-toucan</figcaption></figure>' },
    { template: '<figure class="img-container"><figcaption
class="img-caption">Yellow-warbler</figcaption></figure>' },
    { template: '<figure class="img-container"><figcaption class="img-
caption">Bee-eater</figcaption></figure>' }
  ],
  indicatorsType: "Default",
});
carouselObj.appendTo('#carousel');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.cs
s" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<!--style reference from app-->
<link href="styles.css" rel="stylesheet">
<!--system js reference and configuration-->
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="control-section">
            <div class="control-container">
                <div id="carousel"></div>
            </div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Dynamic Indicator

A dynamic indicator in a carousel provides visual cues or markers that dynamically change or update to indicate the current position. The Dynamic indicator can be achieved by setting the [indicatorsType](#) to **Dynamic**.

INDEX.JS

```

var carouselObj = new ej.navigations.Carousel({
    items: [
        { template: '<figure class="img-container"><figcaption class="img-
caption">Cardinal</figcaption></figure>' },
        { template: '<figure class="img-container"><figcaption class="img-
caption">Kingfisher</figcaption></figure>' },
        { template: '<figure class="img-container"><figcaption
class="img-caption">Keel-billed-toucan</figcaption></figure>' },
        { template: '<figure class="img-container"><figcaption
class="img-caption">Yellow-warbler</figcaption></figure>' },
        { template: '<figure class="img-container"><figcaption class="img-
caption">Bee-eater</figcaption></figure>' }
    ],
    indicatorsType: "Dynamic",
});
carouselObj.appendTo('#carousel');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.cs
s" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="control-section">
      <div class="control-container">
        <div id="carousel"></div>
      </div>
    </div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Fraction Indicator

The fraction indicator type displays the current slide index and total slide count as a fraction. The Fraction indicator can be achieved by setting the [indicatorsType](#) to Fraction.

INDEX.JS

```

var carouselObj = new ej.navigations.Carousel({
  items: [
    { template: '<figure class="img-container"><figcaption class="img-
caption">Cardinal</figcaption></figure>' },
    { template: '<figure class="img-container"><figcaption class="img-
caption">Kingfisher</figcaption></figure>' },
    { template: '<figure class="img-container"><figcaption
class="img-caption">Keel-billed-toucan</figcaption></figure>' },
    { template: '<figure class="img-container"><figcaption
class="img-caption">Yellow-warbler</figcaption></figure>' },
    { template: '<figure class="img-container"><figcaption class="img-
caption">Bee-eater</figcaption></figure>' }
  ],
  indicatorsType: "Fraction",
});
carouselObj.appendTo('#carousel');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link
href="https://cdn.jsdelivrivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.cs
s" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>

```

```

</head>
<body>

    <div id="container">
        <div class="control-section">
            <div class="control-container">
                <div id="carousel"></div>
            </div>
        </div>
    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Progress Indicator

The Progress Indicator type displays the current slide as a progress bar. The Progress indicator can be achieved by setting the [indicatorsType](#) to **Progress**.

INDEX.JS

```

var carouselObj = new ej.navigations.Carousel({
    items: [
        { template: '<figure class="img-container"><figcaption class="img-
caption">Cardinal</figcaption></figure>' },
        { template: '<figure class="img-container"><figcaption class="img-
caption">Kingfisher</figcaption></figure>' },
        { template: '<figure class="img-container"><figcaption
class="img-caption">Keel-billed-toucan</figcaption></figure>' },
        { template: '<figure class="img-container"><figcaption
class="img-caption">Yellow-warbler</figcaption></figure>' },
        { template: '<figure class="img-container"><figcaption class="img-
caption">Bee-eater</figcaption></figure>' }
    ],
    indicatorsType: "Progress",
});
carouselObj.appendTo('#carousel');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>

```

```

<title>Essential JS 2</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
<meta name="description" content="Essential JS 2">
<meta name="author" content="Syncfusion">
<link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.cs
s" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<!--style reference from app-->
<link href="styles.css" rel="stylesheet">
<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="control-section">
            <div class="control-container">
                <div id="carousel"></div>
            </div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Play button

Show or hide the play button

In the carousel, [autoPlay](#) actions have been controlled by using the [showPlayButton](#) property in the user interface. When you enable this property, the slide transitions are controlled using this play and pause button. The following example depicts the code to show the play button in the carousel.

INDEX.JS

```

var carouselObj = new ej.navigations.Carousel({
    items: [
        { template: '<figure class="img-container"><figcaption class="img-
caption">Cardinal</figcaption></figure>' },
    { template: '<figure class="img-container"><figcaption class="img-
caption">Kingfisher</figcaption></figure>' },
    { template: '<figure class="img-container"><figcaption
class="img-caption">Keel-billed-toucan</figcaption></figure>' },
    { template: '<figure class="img-container"><figcaption
class="img-caption">Yellow-warbler</figcaption></figure>' },
    { template: '<figure class="img-container"><figcaption class="img-
caption">Bee-eater</figcaption></figure>' }
    ],
    showPlayButton: true
});
carouselObj.appendTo('#carousel');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.cs
s" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="control-section">
      <div class="control-container">

```



```

        <div id="carousel"></div>
    </div>
</div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Play button template

Template option is provided to customize the play button by using the [playButtonTemplate](#) property. The following example depicts the code for applying a template to play button in the carousel.

INDEX.JS

```

var carouselObj = new ej.navigations.Carousel({
    items: [
        { template: '<figure class="img-container"><figcaption class="img-
caption">Cardinal</figcaption></figure>' },
        { template: '<figure class="img-container"><figcaption class="img-
caption">Kingfisher</figcaption></figure>' },
        { template: '<figure class="img-container"><figcaption
class="img-caption">Keel-billed-toucan</figcaption></figure>' },
        { template: '<figure class="img-container"><figcaption
class="img-caption">Yellow-warbler</figcaption></figure>' },
        { template: '<figure class="img-container"><figcaption class="img-
caption">Bee-eater</figcaption></figure>' }
    ],
    showPlayButton: true,
    playButtonTemplate: '<div id="play"></div>'
});
carouselObj.appendTo('#carousel');
var button = new ej.buttons.Button({ cssClass: 'e-info', content: "Pause"
});
button.appendTo('#play');
button.element.onclick = function () {
    if (carouselObj.autoPlay) {
        button.content = "Play";
        carouselObj.autoPlay = false;
    } else {
        button.content = "Pause";
        carouselObj.autoPlay = true;
    }
}

```

```

    }
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.cs
s" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="control-section">
      <div id="prev"></div>
      <div id="next"></div>
      <div class="control-container">
        <div id="carousel"></div>
      </div>
    </div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Animations and transitions in ##Platform_Name## Carousel control

Animations

Fade animation

In Carousel, two built-in animations are provided for slide transitions. You can disable animation using the [animationEffect](#) property. By default, Slide animation is applied for the transition between slides.

The following demo depicts the example for fade animation,

INDEX.JS

```
var carouselObj = new ej.navigations.Carousel({
  items: [
    { template: '<figure class="img-container"><figcaption class="img-
caption">Cardinal</figcaption></figure>' },
    { template: '<figure class="img-container"><figcaption class="img-
caption">Kingfisher</figcaption></figure>' },
    { template: '<figure class="img-container"><figcaption
class="img-caption">Keel-billed-toucan</figcaption></figure>' },
    { template: '<figure class="img-container"><figcaption
class="img-caption">Yellow-warbler</figcaption></figure>' },
    { template: '<figure class="img-container"><figcaption class="img-
caption">Bee-eater</figcaption></figure>' }
  ],
  animationEffect: 'Fade'
});
carouselObj.appendTo('#carousel');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link
href="https://cdn.jsdelivrivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.cs
s" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
```

```

<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="control-section">
            <div class="control-container">
                <div id="carousel"></div>
            </div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Custom animation

In Carousel, you can use customized animation effects for slide transitions using the [Custom](#) option of the [animationEffect](#) property and apply custom animation css via [cssClass](#) property.

The following demo depicts the example for **parallax** custom animation,

INDEX.JS

```

var carouselObj = new ej.navigations.Carousel({
    items: [
        { template: '<figure class="img-container"><figcaption class="img-
caption">Cardinal</figcaption></figure>' },
        { template: '<figure class="img-container"><figcaption class="img-
caption">Kingfisher</figcaption></figure>' },
        { template: '<figure class="img-container"><figcaption
class="img-caption">Keel-billed-toucan</figcaption></figure>' },
        { template: '<figure class="img-container"><figcaption
class="img-caption">Yellow-warbler</figcaption></figure>' },
        { template: '<figure class="img-container"><figcaption class="img-
caption">Bee-eater</figcaption></figure>' }
    ]
});

```

```

    ],
    cssClass: 'parallax',
    animationEffect: 'Custom'
});
carouselObj.appendTo('#carousel');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.cs
s" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="control-section">
      <div class="control-container">
        <div id="carousel"></div>
      </div>
    </div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Intervals between slides

Using the items property, you can set different intervals for each item to transition between slides. The default interval is 5000 ms (5 seconds). The following example depicts the code for setting the different intervals between each item.

INDEX.JS

```
var carouselObj = new ej.navigations.Carousel({
  items: [
    { template: '<figure class="img-container"><figcaption class="img-
caption">Cardinal</figcaption></figure>', interval: 3000 },
    { template: '<figure class="img-container"><figcaption class="img-
caption">Kingfisher</figcaption></figure>', interval: 1000 },
    { template: '<figure class="img-container"><figcaption
class="img-caption">Keel-billed-toucan</figcaption></figure>', interval:
2000 },
    { template: '<figure class="img-container"><figcaption
class="img-caption">Yellow-warbler</figcaption></figure>', interval: 5000 },
    { template: '<figure class="img-container"><figcaption class="img-
caption">Bee-eater</figcaption></figure>', interval: 6000 }
  ]
});
carouselObj.appendTo('#carousel');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.cs
s" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="control-section">
            <div class="control-container">
                <div id="carousel"></div>
            </div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Note: Interval property can accept value in terms of milliseconds.

Auto play slides

In the carousel, all slides transitions are performed continuously after the specified or default intervals. You can enable or disable the auto slide transition using the [autoPlay](#) property. The following example depicts the code to enable or disable the auto slide transitions.

INDEX.JS

```

var carouselObj = new ej.navigations.Carousel({
    items: [
        { template: '<figure class="img-container"><figcaption class="img-
caption">Cardinal</figcaption></figure>' },
        { template: '<figure class="img-container"><figcaption class="img-
caption">Kingfisher</figcaption></figure>' },
        { template: '<figure class="img-container"><figcaption
class="img-caption">Keel-billed-toucan</figcaption></figure>' },
        { template: '<figure class="img-container"><figcaption
class="img-caption">Yellow-warbler</figcaption></figure>' },
        { template: '<figure class="img-container"><figcaption class="img-
caption">Bee-eater</figcaption></figure>' }
    ],

```

```

    autoPlay: true
  });
  carouselObj.appendTo('#carousel');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.cs
s" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="control-section">
      <div class="control-container">
        <div id="carousel"></div>
      </div>
    </div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Pause on hover

By default, Slide transitions are paused when hovering the mouse pointer over the Carousel element. You can enable or disable this functionality using the [pauseOnHover](#) property.

The following example depicts the code to play the slides when hovering the mouse pointer over the Carousel element.

INDEX.JS

```
var carouselObj = new ej.navigations.Carousel({
  items: [
    { template: '<figure class="img-container"><figcaption class="img-
caption">Cardinal</figcaption></figure>' },
    { template: '<figure class="img-container"><figcaption class="img-
caption">Kingfisher</figcaption></figure>' },
    { template: '<figure class="img-container"><figcaption
class="img-caption">Keel-billed-toucan</figcaption></figure>' },
    { template: '<figure class="img-container"><figcaption
class="img-caption">Yellow-warbler</figcaption></figure>' },
    { template: '<figure class="img-container"><figcaption class="img-
caption">Bee-eater</figcaption></figure>' }
  ],
  pauseOnHover: false
});
carouselObj.appendTo('#carousel');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link
href="https://cdn.jsdelivrivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.cs
s" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="control-section">
            <div class="control-container">
                <div id="carousel"></div>
            </div>
        </div>
    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Looping slides

In the carousel, slides transitions are repeated continuously when you reach the last slide by default. You can enable or disable the infinite slide transition using the [loop](#) property. The following example depicts the code to enable or disable the infinite slide transitions.

INDEX.JS

```

var carouselObj = new ej.navigations.Carousel({
    items: [
        { template: '<figure class="img-container"><figcaption class="img-
caption">Cardinal</figcaption></figure>' },
        { template: '<figure class="img-container"><figcaption class="img-
caption">Kingfisher</figcaption></figure>' },
        { template: '<figure class="img-container"><figcaption
class="img-caption">Keel-billed-toucan</figcaption></figure>' },
        { template: '<figure class="img-container"><figcaption
class="img-caption">Yellow-warbler</figcaption></figure>' },
        { template: '<figure class="img-container"><figcaption class="img-
caption">Bee-eater</figcaption></figure>' }
    ],
    loop: true
});
carouselObj.appendTo('#carousel');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.cs
s" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="control-section">
      <div class="control-container">
        <div id="carousel"></div>
      </div>
    </div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Slide changing events

Using the [slideChanging](#) or [slideChanged](#) events of the Carousel component, you can perform sample end customization while the carousel items are switched.

The following demo depicts the example for carousel events,

INDEX.JS

```

var carouselObj = new ej.navigations.Carousel({

```

```

    items: [
      { template: '<figure class="img-container"><figcaption class="img-
caption">Cardinal</figcaption></figure>' },
      { template: '<figure class="img-container"><figcaption class="img-
caption">Kingfisher</figcaption></figure>' },
      { template: '<figure class="img-container"><figcaption
class="img-caption">Keel-billed-toucan</figcaption></figure>' },
      { template: '<figure class="img-container"><figcaption
class="img-caption">Yellow-warbler</figcaption></figure>' },
      { template: '<figure class="img-container"><figcaption class="img-
caption">Bee-eater</figcaption></figure>' }
    ],
    slideChanging: function (args) {
      args.currentSlide; // You can customize the slide before changing.
    },
    slideChanged: function (args) {
      args.currentSlide; // You can customize the slide after changed.
    }
  });
carouselObj.appendTo('#carousel');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.cs
s" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>

```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="control-section">
            <div class="control-container">
                <div id="carousel"></div>
            </div>
        </div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Disable touch swiping

In the carousel, you can swipe the carousel slides using touch actions by default. The swipe action can be enabled or disabled using the [enableTouchSwipe](#) property. The following example depicts the code to disable the swipe action for the slide.

INDEX.JS

```

var carouselObj = new ej.navigations.Carousel({
    items: [
        { template: '<figure class="img-container"><figcaption class="img-
caption">Cardinal</figcaption></figure>' },
        { template: '<figure class="img-container"><figcaption class="img-
caption">Kingfisher</figcaption></figure>' },
        { template: '<figure class="img-container"><figcaption
class="img-caption">Keel-billed-toucan</figcaption></figure>' },
        { template: '<figure class="img-container"><figcaption
class="img-caption">Yellow-warbler</figcaption></figure>' },
        { template: '<figure class="img-container"><figcaption class="img-
caption">Bee-eater</figcaption></figure>' }
    ],
    enableTouchSwipe: false
});
carouselObj.appendTo('#carousel');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.cs
s" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="control-section">
      <div class="control-container">
        <div id="carousel"></div>
      </div>
    </div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Swipe Modes

In the carousel, the [swipeMode](#) property allows specifying whether the slide transition should occur while performing swiping via touch or mouse. The slide swiping is enabled or disabled using the bitwise operator.

The following are the different swipe modes available in the carousel:

- CarouselSwipeMode.Touch - Allows the user to slide the slides using touch actions.
- CarouselSwipeMode.Mouse - Allows the user to slide the slides using mouse actions.

- CarouselSwipeMode.Touch & CarouselSwipeMode.Mouse - Allows the user to slide the slides using both touch and mouse actions.
- ~CarouselSwipeMode.Touch & ~CarouselSwipeMode.Mouse - Disables both touch and mouse actions.

INDEX.JS

```
var carouselObj = new ej.navigations.Carousel({
  items: [
    { template: '<figure class="img-container"><figcaption class="img-
caption">Cardinal</figcaption></figure>' },
    { template: '<figure class="img-container"><figcaption class="img-
caption">Kingfisher</figcaption></figure>' },
    { template: '<figure class="img-container"><figcaption
class="img-caption">Keel-billed-toucan</figcaption></figure>' },
    { template: '<figure class="img-container"><figcaption
class="img-caption">Yellow-warbler</figcaption></figure>' },
    { template: '<figure class="img-container"><figcaption class="img-
caption">Bee-eater</figcaption></figure>' }
  ],
  swipeMode: ej.navigations.CarouselSwipeMode.Mouse &
ej.navigations.CarouselSwipeMode.Touch
});
carouselObj.appendTo('#carousel');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.cs
s" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
```

```

<!--system js reference and configuration-->
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="control-section">
            <div class="control-container">
                <div id="carousel"></div>
            </div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Accessibility in ##Platform_Name## Carousel control

The accessibility compliance for the Carousel control is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support | |

| [Section 508](#) Support | |

| Screen Reader Support | |

| Right-To-Left Support | |

| Color Contrast | |

| Mobile Device Support | |

| Keyboard Navigation Support | |

| [Accessibility Checker](#) Validation | |

| [Axe-core](#) Accessibility Validation | |


```
<style>
.post .post-content img {
display: inline-block;
margin: 0.5em 0;
}
</style>

<div> - All
features of the control meet the requirement.</div>

<div> - Some features of the control do not meet the requirement.</div>

<div> - The
control does not meet the requirement.</div>
```

ARIA attributes

The Carousel control is designed by considering [WAI-ARIA](#) standard. Carousel is supported with ARIA Accessibility which is accessible by on-screen readers and other assistive technology devices. The following list of attributes is added to the Carousel.

| Roles and Attributes | Functionalities

----- -----	

aria-roledescription	The role description attribute has been added for the root element (carousel) and each carousel slide item (slide).
aria-label	Previous, next and play/pause buttons and all indicator elements.
aria-current	For the active item indicator element, <code>aria-current</code> is set to <code>true</code> .
aria-hidden	For all carousel elements except the currently visible item, <code>aria-hidden</code> is set to <code>true</code> .
aria-live	For carousel items element, when <code>autoPlay</code> is <code>true</code> , <code>aria-live</code> is set to <code>off</code> ; when <code>autoPlay</code> is <code>false</code> , <code>aria-live</code> is set to <code>polite</code> .
aria-role	For carousel slide item, <code>aria-role</code> has been grouped.

Keyboard interaction

By default, keyboard navigation is enabled. This control implements keyboard navigation support by following the WAI-ARIA practices. Once focused on the active Carousel element, you can use the following key combination for interacting with the Carousel.

Key	Description
-----	-----

Alt + J	Keys to focus the Carousel control (done at application end).	
Arrows	Keys to navigate between slides.	
Home	To navigate to the first slide.	
End	To navigate to the last slide.	
Space	To play/pause the slide transitions.	
Enter	To perform the respective action on its focus.	
Tab	To Move focus through the interactive elements.	
Shift + Tab	To Move focus through the interactive elements.	

Ensuring accessibility

The Carousel control accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Carousel control is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Carousel control with accessibility tools.

See also

- [Accessibility in Syncfusion components](#)

Styles and appearance in ##Platform_Name## Carousel control

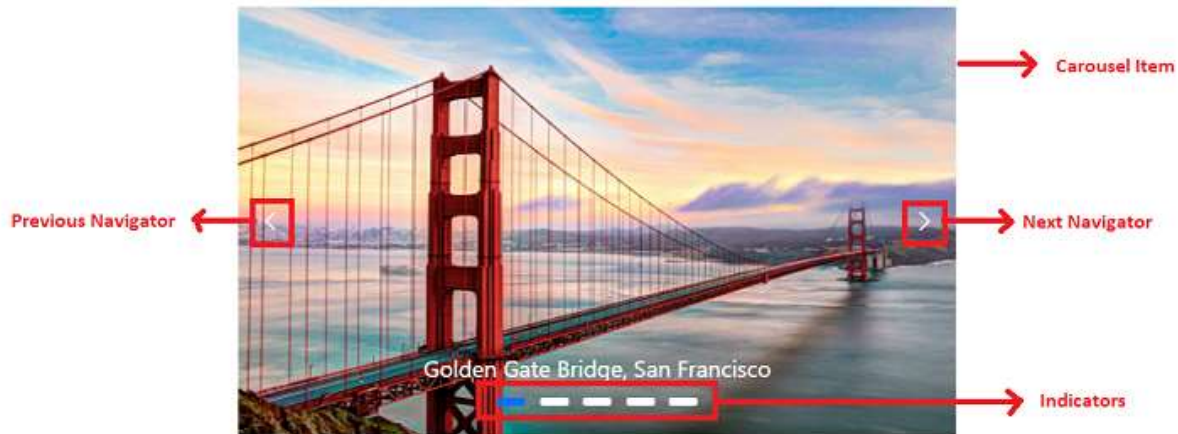
To modify the Carousel appearance, you need to override the default CSS of Carousel component. Please find the list of CSS classes and its corresponding section in Carousel component. Also, you have an option to create your own custom theme for the controls using our [Theme Studio](#).

CSS Structure in JavaScript Carousel Control

The following content provides the exact CSS structure that can be used to modify the control's appearance based on user preference.

CSS Class | Purpose of Class

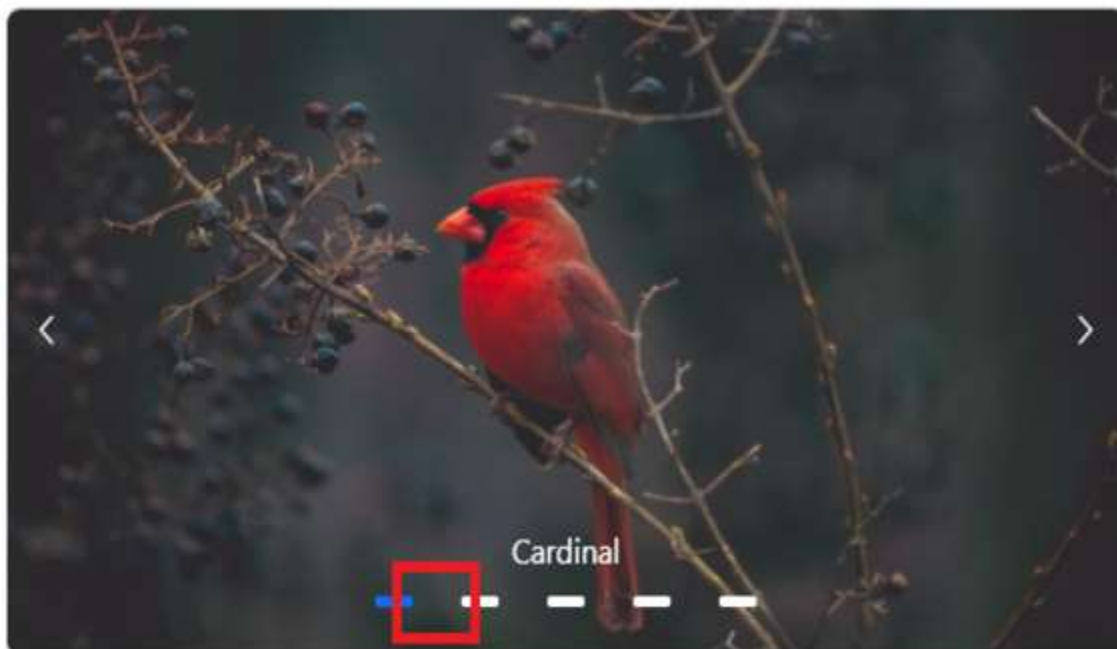
.e-carousel .e-carousel-item	To customize the carousel item
.e-carousel-item.e-active	To customize the active carousel item
.e-carousel .e-carousel-indicators	To customize the indicators
.e-carousel .e-carousel-indicators .e-indicator-bars .e-indicator-bar	To customize the indicator bars
.e-carousel .e-carousel-indicators .e-indicator-bars .e-indicator-bar .e-indicator	To customize the individual indicator appearance
.e-carousel .e-carousel-navigators	To customize the navigators
.e-carousel .e-carousel-navigators .e-previous	To customize the previous button
.e-carousel .e-carousel-navigators .e-next	To customize the next button
.e-carousel .e-carousel-navigators .e-play-pause	To customize the play and pause button
.e-carousel.e-partial .e-carousel-slide-container	To customize the partial visible slides



Customizing the indicators

Use the following CSS to customize the space between indicators by overriding the `.e-indicator-bar` CSS class.

```
`css
.e-carousel .e-carousel-indicators .e-indicator-bars .e-indicator-bar {
padding: 8px;
}
```



Use the following CSS to customize the indicators appearance by overriding the `.e-indicator` CSS class.

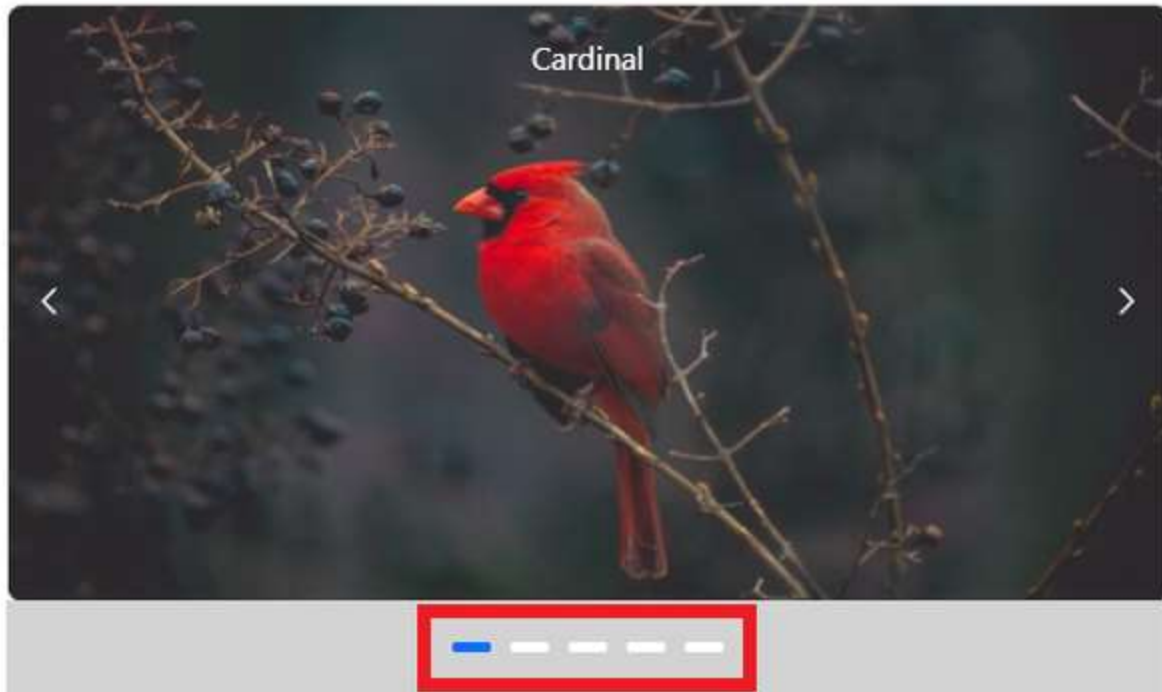
```
`css
```

```
.e-carousel .e-carousel-indicators .e-indicator-bars .e-indicator-bar .e-indicator {  
width: 20px;  
border-radius: 100%;  
}  
`css
```



Use the following CSS to render the indicators outside the carousel items by overriding the `.e-carousel-indicators` CSS class.

```
`css  
.e-carousel .e-carousel-indicators {  
bottom: auto;  
}  
`css
```



Customizing the navigators

Use the following CSS to customize the previous and next icon size and colors.

```
`css
```

```
.e-carousel .e-carousel-navigators .e-next .e-btn:not(:disabled) .e-btn-icon,  
.e-carousel .e-carousel-navigators .e-previous .e-btn:not(:disabled) .e-btn-icon  
{  
  color: greenyellow;  
  font-size: 25px;  
}
```



Use the following CSS to customize the navigators position to bottom by overriding the `.e-carousel-navigators` CSS class.

```
`css
.e-carousel .e-carousel-navigators {
top: 120px;
}
`
```



Use the following CSS to render the previous and next icon to outside the carousel items by overriding the `.e-previous` and `.e-next` CSS class.

```
`css
.e-carousel .e-carousel-navigators .e-previous,
.e-carousel .e-carousel-navigators .e-next
{
margin: -60px;
background: black;
}
`
```




Customizing partial slides size

You can customize the partial slide size by overriding the `.e-carousel-slide-container` CSS class.

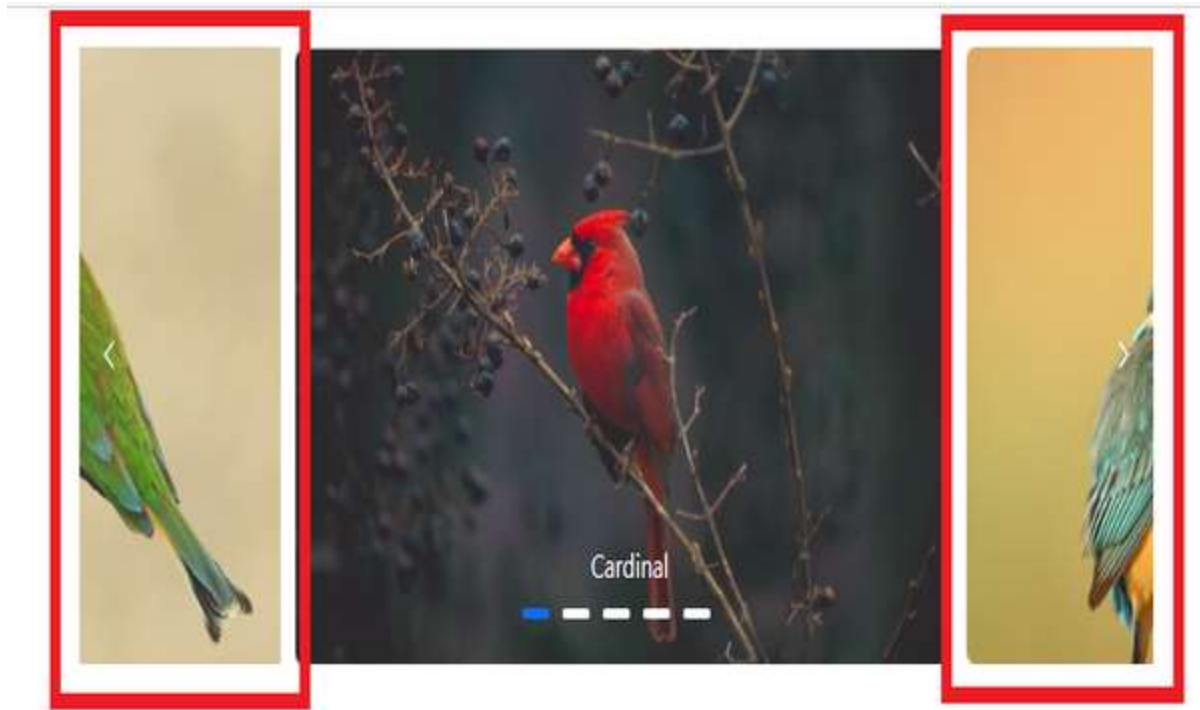
```
`css
```

```
.e-carousel.e-partial .e-carousel-slide-container{
```

```
padding: 0 150px;
```

```
}
```

```
`
```

Chart

<!-- markdownlint-disable MD036 -->

Working with data in ##Platform_Name## Chart control

Chart can visualise data bound from local or remote data.

Local Data

You can bind a simple JSON data to the chart using [dataSource](#) property in series. Now map the fields in JSON to [xName](#) and [yName](#) properties.

INDEX.JS

```
var chartData = [
  { month: 'Jan', sales: 35 }, { month: 'Feb', sales: 28 },
  { month: 'Mar', sales: 34 }, { month: 'Apr', sales: 32 },
  { month: 'May', sales: 40 }, { month: 'Jun', sales: 32 },
  { month: 'Jul', sales: 35 }, { month: 'Aug', sales: 55 },
  { month: 'Sep', sales: 38 }, { month: 'Oct', sales: 30 },
  { month: 'Nov', sales: 25 }, { month: 'Dec', sales: 32 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    valueType: 'Category'
  },
  series: [{
    dataSource: chartData,
    xName: 'month',
    yName: 'sales',
    type: 'Column'
  }]
}, '#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Remote Data

You can also bind remote data to the chart using DataManager. The **DataManager** requires minimal information like webservice URL, adaptor and crossDomain to interact with service endpoint properly. Assign the instance of DataManager to the [dataSource](#) property in series and map the fields of data to [xName](#) and

[yName](#) properties. You can also use the [query](#) property of the series to filter the data.

INDEX.JS

```

var dataManager = new ej.data.DataManager({
  url: 'https://services.syncfusion.com/js/production/api/orders'
});
var query = new ej.data.Query().take(5).where('Estimate', 'lessThan', 3,
false);
var chart = new ej.charts.Chart({
  primaryXAxis: {
    valueType: 'Category'
  },
  primaryYAxis:
  {
    title: 'Freight rate in U.S. dollars'
  },
  series: [

```

```

        {
            type: 'Column',
            dataSource: dataManager,
            xName: 'CustomerID', yName: 'Freight', query: query
        }
    ],
    title: 'Container freight rate'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

Binding Data Using ODataAdaptor

OData is a standardized protocol for creating and consuming data. You can retrieve data from OData service using the DataManager. Refer to the following code example for remote Data binding using OData service.

INDEX.JS

```

var query = new ej.data.Query();
var data = new ej.data.DataManager({
    url: 'https://services.syncfusion.com/js/production/api/orders',
    adaptor: new ej.data.ODataAdaptor()
});
var chart = new ej.charts.Chart({
    //Initializing Primary X Axis
    primaryXAxis: {
        valueType: 'Category',

```

```

    },
    //Initializing Chart Sample
    series: [
        {
            type: 'Column',
            dataSource: data,
            xName: 'CustomerID', yName: 'Freight', query: query,
        }
    ],
    title: 'Sprint Task Analysis'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Lazy loading

Lazy loading allows you to load data for chart on demand. Chart will fire the scrollEnd event, in that we can get the minimum and maximum range of the axis, based on this, we can upload the data to chart.

INDEX.JS

```

var intl = new ej.base.Internationalization();
var chart = new ej.charts.Chart({
  primaryXAxis: {
    title: 'Day',
    valueType: 'DateTime',
    edgeLabelPlacement: 'Shift',
    skeleton: 'yMMM',

```

```

        skeletonType: 'Date',
        scrollbarSettings: {
            range: {
                minimum: new Date(2009, 0, 1),
                maximum: new Date(2014, 0, 1)
            },
            enable: true,
        }
    },
    primaryYAxis: {
        title: 'Server Load',
        labelFormat: '{value}MB'
    },
    series: [{
        dataSource: GetDateTimeData(new Date(2009, 0, 1), new Date(2009, 8,
1)),
        xName: 'x', yName: 'y',
        type: 'Line', animation: { enable: false },
    }],
    height: '450',
    title: 'Network Load',
    crosshair: { enable: true, lineType: 'Vertical' },
    tooltip: { enable: true, shared: true },
    legendSettings: { visible: true },
    scrollEnd: function (args) {
        if (lazymode.value === 'Range') {
            chart.series[0].dataSource =
GetDateTimeData(args.currentRange.minimum, args.currentRange.maximum);
        }
        chart.dataBind();
    },
}, '#element');
function GetDateTimeData(start, end) {
    var series1 = [];
    var date;
    var value = 30;
    var option = {
        skeleton: 'full',
        type: 'dateTime'
    };
    var dateParser = intl.getDateParser(option);
    var dateFormatter = intl.getDateFormat(option);
    for (var i = 0; start <= end; i++) {
        date = Date.parse(dateParser(dateFormatter(start)));
        if (Math.random() > .5) {
            value += (Math.random() * 10 - 5);
        }
        else {
            value -= (Math.random() * 10 - 5);
        }
        if (value < 0) {
            value = getRandomInt(20, 40);
        }
        var point1 = { x: new Date(date), y: Math.round(value) };
        new Date(start.setDate(start.getDate() + 1));
        series1.push(point1);
    }
}

```

```

    return series1;
}
function getRandomInt(min, max) {
    return Math.floor(Math.random() * (max - min + 1)) + min;
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Empty points

The Data points that uses the `null` or `undefined` as value are considered as empty points. Empty data points are ignored and not plotted in the Chart. When the data is provided by using the points property, By using `emptyPointSettings` property in series, you can customize the empty point. Default `mode` of the empty point is `Gap`.

INDEX.JS

```

var chartData = [
  { month: 'Jan', sales: 35 }, { month: 'Feb', sales: 28 },
  { month: 'Mar', sales: null }, { month: 'Apr', sales: 32 },
  { month: 'May', sales: 40 }, { month: 'Jun', sales: 32 },
  { month: 'Jul', sales: 35 }, { month: 'Aug', sales: undefined },
  { month: 'Sep', sales: 38 }, { month: 'Oct', sales: 30 },
  { month: 'Nov', sales: 25 }, { month: 'Dec', sales: 32 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    valueType: 'Category'
  }
});

```

```

    },
    series:[{
        dataSource: chartData,
        xName: 'month',
        yName: 'sales',
        type: 'Column',
        emptyPointSettings: {
            mode: 'Gap'
        }
    },
    {
        dataSource: chartData,
        xName: 'month',
        yName: 'sales',
        type: 'Line',
        marker: { visible: true },
        emptyPointSettings: {
            mode: 'Average'
        }
    }
    ]
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Customizing empty point

Specific color for empty point can be set by **fill** property in **emptyPointSettings**. Border for a empty point can be set by **border** property.

INDEX.JS

```
var chartData = [
    { month: 'Jan', sales: 35 }, { month: 'Feb', sales: 28 },
    { month: 'Mar', sales: null }, { month: 'Apr', sales: 32 },
    { month: 'May', sales: 40 }, { month: 'Jun', sales: 32 },
    { month: 'Jul', sales: 35 }, { month: 'Aug', sales: undefined },
    { month: 'Sep', sales: 38 }, { month: 'Oct', sales: 30 },
    { month: 'Nov', sales: 25 }, { month: 'Dec', sales: 32 }
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        valueType: 'Category'
    },
    series:[{
        dataSource: chartData,
        xName: 'month',
        yName: 'sales',
        type: 'Column',
        emptyPointSettings: {
            mode: 'Average',
            fill: 'green',
            border: { color: 'black', width: 2}
        }
    },
    {
        dataSource: chartData,
        xName: 'month',
        yName: 'sales',
        type: 'Line',
        marker: { visible: true},
        emptyPointSettings: {
            mode: 'Zero',
            fill: 'pink'
        }
    }
    ]
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
```



```

</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Chart dimensions in ##Platform_Name## Chart control

Size for Container

Chart can render to its container size. You can set the size via inline or CSS as demonstrated below.

```

<div id='container'>

<div id='element' style="width:650px; height:350px;"></div>

</div>

```

INDEX.JS

```

var chartData = [
    { month: 'Jan', sales: 35 }, { month: 'Feb', sales: 28 },
    { month: 'Mar', sales: 34 }, { month: 'Apr', sales: 32 },
    { month: 'May', sales: 40 }, { month: 'Jun', sales: 32 },
    { month: 'Jul', sales: 35 }, { month: 'Aug', sales: 55 },
    { month: 'Sep', sales: 38 }, { month: 'Oct', sales: 30 },
    { month: 'Nov', sales: 25 }, { month: 'Dec', sales: 32 }
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        valueType: 'Category'
    },
    series:[{
        dataSource: chartData,
        xName: 'month',
        yName: 'sales',
        type: 'Line'
    }],
    // Width and height for chart in pixel
    width: '650', height: '350'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">

```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

<!-- markdownlint-disable MD036 -->

Size for Chart

<!-- markdownlint-disable MD036 -->

You can also set size for chart directly through [width](#) and [height](#) properties.

In Pixel

You can set the size of chart in pixel as demonstrated below.

INDEX.JS

```

var chartData = [
    { month: 'Jan', sales: 35 }, { month: 'Feb', sales: 28 },
    { month: 'Mar', sales: 34 }, { month: 'Apr', sales: 32 },
    { month: 'May', sales: 40 }, { month: 'Jun', sales: 32 },
    { month: 'Jul', sales: 35 }, { month: 'Aug', sales: 55 },
    { month: 'Sep', sales: 38 }, { month: 'Oct', sales: 30 },
    { month: 'Nov', sales: 25 }, { month: 'Dec', sales: 32 }
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        valueType: 'Category'
    },
    series:[{
        dataSource: chartData,
        xName: 'month',
        yName: 'sales',
        type: 'Line'
    }],
    // Width and height for chart in pixel

```

```
width: '650', height: '350'
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

In Percentage

By setting value in percentage, chart gets its dimension with respect to its container. For example, when the height is '50%', chart renders to half of the container height.

INDEX.JS

```
var chartData = [
  { month: 'Jan', sales: 35 }, { month: 'Feb', sales: 28 },
  { month: 'Mar', sales: 34 }, { month: 'Apr', sales: 32 },
  { month: 'May', sales: 40 }, { month: 'Jun', sales: 32 },
  { month: 'Jul', sales: 35 }, { month: 'Aug', sales: 55 },
  { month: 'Sep', sales: 38 }, { month: 'Oct', sales: 30 },
  { month: 'Nov', sales: 25 }, { month: 'Dec', sales: 32 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    valueType: 'Category'
  },
  series:[{
    dataSource: chartData,
    xName: 'month',
    yName: 'sales',
```

```

        type: 'Line'
    }],
    // Width and height for chart in percentage
    width: '80%', height: '90%'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Note: When you do not specify the size, it takes 450px as the height and window size as its width.

Category axis in ##Platform_Name## Chart control

<!-- markdownlint-disable MD036 -->

Category axis are used to represent, the string values instead of numbers.

INDEX.JS

```

var chartData = [
  { country: "USA", gold: 50, silver: 70, bronze: 45 },
  { country: "China", gold: 40, silver: 60, bronze: 55 },
  { country: "Japan", gold: 70, silver: 60, bronze: 50 },
  { country: "Australia", gold: 60, silver: 56, bronze: 40 },
  { country: "France", gold: 50, silver: 45, bronze: 35 },
  { country: "Germany", gold: 40, silver: 30, bronze: 22 },
  { country: "Italy", gold: 40, silver: 35, bronze: 37 },
  { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart = new ej.charts.Chart({

```

```

primaryXAxis: {
    //Category in primary X Axis
    valueType: 'Category',
    title: 'Countries'
},
primaryYAxis: {
    minimum: 0, maximum: 80,
    interval: 20, title: 'Medals'
},
series:[{
    dataSource: chartData,
    xName: 'country', yName: 'gold',
    name: 'Gold', type: 'Column'
}, {
    dataSource: chartData,
    xName: 'country', yName: 'silver',
    name: 'Silver', type: 'Column'
}, {
    dataSource: chartData,
    xName: 'country', yName: 'bronze',
    name: 'Bronze', type: 'Column'
}],
title: 'Olympic Medals'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Note: To use category axis, we need to inject `Category` module using `Chart.Inject(Category)` method and set the `valueType` of axis to `Category`.

Labels Placement

By default, category labels are placed between the ticks in an axis, this can also be placed on ticks using `[labelPlacement](../api/chart/axis/)` property.

INDEX.JS

```
var chartData = [
  { country: "USA", gold: 50, silver: 70, bronze: 45 },
  { country: "China", gold: 40, silver: 60, bronze: 55 },
  { country: "Japan", gold: 70, silver: 60, bronze: 50 },
  { country: "Australia", gold: 60, silver: 56, bronze: 40 },
  { country: "France", gold: 50, silver: 45, bronze: 35 },
  { country: "Germany", gold: 40, silver: 30, bronze: 22 },
  { country: "Italy", gold: 40, silver: 35, bronze: 37 },
  { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    valueType: 'Category',
    // label placement as on ticks
    labelPlacement: 'OnTicks',
    title: 'Countries'
  },
  primaryYAxis: {
    minimum: 0, maximum: 80,
    interval: 20, title: 'Medals'
  },
  series:[{
    dataSource: chartData,
    xName: 'country', yName: 'gold',
    name: 'Gold', type: 'Column'
  }, {
    dataSource: chartData,
    xName: 'country', yName: 'silver',
    name: 'Silver', type: 'Column'
  }, {
    dataSource: chartData,
    xName: 'country', yName: 'bronze',
    name: 'Bronze', type: 'Column'
  }],
  title: 'Olympic Medals'
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Range

Range of the category axis can be customized using [minimum](#), [maximum](#) and [interval](#) property of the axis.

INDEX.JS

```

var chartData = [
    { country: "USA", gold: 50, silver: 70, bronze: 45 },
    { country: "China", gold: 40, silver: 60, bronze: 55 },
    { country: "Japan", gold: 70, silver: 60, bronze: 50 },
    { country: "Australia", gold: 60, silver: 56, bronze: 40 },
    { country: "France", gold: 50, silver: 45, bronze: 35 },
    { country: "Germany", gold: 40, silver: 30, bronze: 22 },
    { country: "Italy", gold: 40, silver: 35, bronze: 37 },
    { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        valueType: 'Category',
        // label placement as on ticks
        labelPlacement: 'OnTicks',
        title: 'Countries'
    },
    primaryYAxis: {
        minimum: 0, maximum: 80,
        interval: 20, title: 'Medals'
    },
    series:[{
        dataSource: chartData,
        xName: 'country', yName: 'gold',
        name: 'Gold', type: 'Column'
    }, {
        dataSource: chartData,
        xName: 'country', yName: 'silver',
        name: 'Silver', type: 'Column'
    }, {

```

```

        dataSource: chartData,
        xName: 'country', yName: 'bronze',
        name: 'Bronze', type: 'Column'
    }],
    title: 'Olympic Medals'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Indexed category axis

Category axis also can be rendered based on the index values of data source. This can be achieved by defining the `isIndexed` property to `true` in the axis.

INDEX.JS

```

var chartData = [
  { country: "USA", gold: 50, silver: 70, bronze: 45 },
  { country: "China", gold: 40, silver: 60, bronze: 55 },
  { country: "Japan", gold: 70, silver: 60, bronze: 50 },
  { country: "Australia", gold: 60, silver: 56, bronze: 40 },
  { country: "France", gold: 50, silver: 45, bronze: 35 },
  { country: "Germany", gold: 40, silver: 30, bronze: 22 },
  { country: "Italy", gold: 40, silver: 35, bronze: 37 },
  { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {

```



```

        valueType: 'Category',
        // label placement as on ticks
        labelPlacement: 'OnTicks',
        title: 'Countries'
    },
    primaryYAxis: {
        minimum: 0, maximum: 80,
        interval: 20, title: 'Medals'
    },
    series:[{
        dataSource: chartData,
        xName: 'country', yName: 'gold',
        name: 'Gold', type: 'Column'
    }, {
        dataSource: chartData,
        xName: 'country', yName: 'silver',
        name: 'Silver', type: 'Column'
    }, {
        dataSource: chartData,
        xName: 'country', yName: 'bronze',
        name: 'Bronze', type: 'Column'
    }],
    title: 'Olympic Medals'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

```
<!-- markdownlint-disable MD036 -->
```

Numeric axis in ##Platform_Name## Chart control

INDEX.JS

```
var chartData = [
  { x: 1, y: 7 }, { x: 2, y: 1 }, { x: 3, y: 1 },
  { x: 4, y: 14 }, { x: 5, y: 1 }, { x: 6, y: 10 },
  { x: 7, y: 8 }, { x: 8, y: 6 }, { x: 9, y: 10 },
  { x: 10, y: 10 }, { x: 11, y: 16 }, { x: 12, y: 6 },
  { x: 13, y: 14 }, { x: 14, y: 7 }, { x: 15, y: 5 },
  { x: 16, y: 2 }, { x: 17, y: 14 }, { x: 18, y: 7 },
  { x: 19, y: 7 }, { x: 20, y: 10 }];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    // Numerical scale in primary X Axis
    valueType: 'Double',
    title: 'Overs'
  },
  primaryYAxis: {
    title: 'Runs'
  },
  series:[{
    dataSource: chartData,
    xName: 'x', yName: 'y',
    name: 'England', type: 'Area'
  }],
  title: 'England - Run Rate'
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
```

```

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Range

Range of an axis, will be calculated automatically based on the provided data, you can also customize the range of the axis using [minimum](#), [maximum](#) and [interval](#) property of the axis.

INDEX.JS

```

var chartData = [
  { x: 1, y: 7 }, { x: 2, y: 1 }, { x: 3, y: 1 },
  { x: 4, y: 14 }, { x: 5, y: 1 }, { x: 6, y: 10 },
  { x: 7, y: 8 }, { x: 8, y: 6 }, { x: 9, y: 10 },
  { x: 10, y: 10 }, { x: 11, y: 16 }, { x: 12, y: 6 },
  { x: 13, y: 14 }, { x: 14, y: 7 }, { x: 15, y: 5 },
  { x: 16, y: 2 }, { x: 17, y: 14 }, { x: 18, y: 7 },
  { x: 19, y: 7 }, { x: 20, y: 10 }];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    valueType: 'Double',
    // Numeric axis range
    minimum: 1,
    maximum: 20,
    interval: 5,
    title: 'Overs'
  },
  primaryYAxis: {
    title: 'Runs',
    minimum: 0,
    maximum: 20,
    interval: 10
  },
  series: [{
    dataSource: chartData,
    xName: 'x', yName: 'y',
    name: 'England', type: 'Area'
  }],
  title: 'England - Run Rate'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>

```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Range Padding

Padding can be applied to the minimum and maximum extremes of an axis range by using the [rangePadding](#) property. Numeric axis supports the following types of padding.

- None
- Round
- Additional
- Normal
- Auto

Numeric - None

When the [rangePadding](#) is set to **None**, minimum and maximum of the axis is based on the data.

INDEX.JS

```

var chartData = [
    { x: 1, y: 7 }, { x: 2, y: 1 }, { x: 3, y: 1 },
    { x: 4, y: 14 }, { x: 5, y: 1 }, { x: 6, y: 10 },
    { x: 7, y: 8 }, { x: 8, y: 6 }, { x: 9, y: 10 },
    { x: 10, y: 10 }, { x: 11, y: 16 }, { x: 12, y: 6 },
    { x: 13, y: 14 }, { x: 14, y: 7 }, { x: 15, y: 5 },
    { x: 16, y: 2 }, { x: 17, y: 14 }, { x: 18, y: 7 },
    { x: 19, y: 7 }, { x: 20, y: 10 }];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        valueType: 'Double',
        title: 'Overs'
    },
    primaryYAxis: {
        valueType: 'Double',
        title: 'Runs',
        //RangePadding as none in Y Axis
        rangePadding: 'None'
    },
    series:[{
        dataSource: chartData,

```

```

        xName: 'x', yName: 'y', width: 3,
        name: 'England', type: 'Line'
    }},
    title: 'England - Run Rate'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Numeric - Round

When the [rangePadding](#) is set to **Round**, minimum and maximum will be rounded to the nearest possible value, which is divisible by interval. For example, when the minimum is 3.5 and the interval is 1, then the minimum will be rounded to 3.

INDEX.JS

```

var chartData = [
  { x: 1, y: 7 }, { x: 2, y: 1 }, { x: 3, y: 1 },
  { x: 4, y: 14 }, { x: 5, y: 1 }, { x: 6, y: 10 },
  { x: 7, y: 8 }, { x: 8, y: 6 }, { x: 9, y: 10 },
  { x: 10, y: 10 }, { x: 11, y: 16 }, { x: 12, y: 6 },
  { x: 13, y: 14 }, { x: 14, y: 7 }, { x: 15, y: 5 },
  { x: 16, y: 2 }, { x: 17, y: 14 }, { x: 18, y: 7 },
  { x: 19, y: 7 }, { x: 20, y: 10 }];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    valueType: 'Double',
    title: 'Overs'
  }
});

```

```

    },
    primaryYAxis: {
        valueType: 'Double',
        title: 'Runs',
        //RangePadding as round in Y Axis
        rangePadding: 'Round'
    },
    series:[{
        dataSource: chartData,
        xName: 'x', yName: 'y', width: 3,
        name: 'England', type: 'Line'
    }],
    title: 'England - Run Rate'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Numeric - Additional

When the [rangePadding](#) is set to **Additional**, interval of an axis will be added to the minimum and maximum of the axis.

INDEX.JS

```

var chartData = [
  { x: 1, y: 7 }, { x: 2, y: 1 }, { x: 3, y: 1 },
  { x: 4, y: 14 }, { x: 5, y: 1 }, { x: 6, y: 10 },
  { x: 7, y: 8 }, { x: 8, y: 6 }, { x: 9, y: 10 },

```

```

    { x: 10, y: 10 }, { x: 11, y: 16 }, { x: 12, y: 6 },
    { x: 13, y: 14 }, { x: 14, y: 7 }, { x: 15, y: 5 },
    { x: 16, y: 2 }, { x: 17, y: 14 }, { x: 18, y: 7 },
    { x: 19, y: 7 }, { x: 20, y: 10 }]];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    valueType: 'Double',
    title: 'Overs'
  },
  primaryYAxis: {
    valueType: 'Double',
    title: 'Runs',
    //RangePadding as additional in Y Axis
    rangePadding: 'Additional'
  },
  series:[{
    dataSource: chartData,
    xName: 'x', yName: 'y', width: 3,
    name: 'England', type: 'Line'
  }],
  title: 'England - Run Rate'
}, '#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Numeric - Normal

When the [rangePadding](#) is set to **Normal**, padding is applied to the axis based on default range calculation.

INDEX.JS

```
var chartData = [
  { x: 1, y: 7 }, { x: 2, y: 1 }, { x: 3, y: 1 },
  { x: 4, y: 14 }, { x: 5, y: 1 }, { x: 6, y: 10 },
  { x: 7, y: 8 }, { x: 8, y: 6 }, { x: 9, y: 10 },
  { x: 10, y: 10 }, { x: 11, y: 16 }, { x: 12, y: 6 },
  { x: 13, y: 14 }, { x: 14, y: 7 }, { x: 15, y: 5 },
  { x: 16, y: 2 }, { x: 17, y: 14 }, { x: 18, y: 7 },
  { x: 19, y: 7 }, { x: 20, y: 10 }];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    valueType: 'Double',
    title: 'Overs'
  },
  primaryYAxis: {
    valueType: 'Double',
    title: 'Runs',
    //RangePadding as additional in Y Axis
    rangePadding: 'Additional'
  },
  series:[{
    dataSource: chartData,
    xName: 'x', yName: 'y', width: 3,
    name: 'England', type: 'Line'
  }],
  title: 'England - Run Rate'
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
```



```

    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Numeric - Auto

When the [rangePadding](#) is set to **Auto**, horizontal numeric axis takes None as padding calculation, while the vertical numeric axis takes Normal as padding calculation.

INDEX.JS

```

var chartData = [
    { x: 1, y: 7 }, { x: 2, y: 1 }, { x: 3, y: 1 },
    { x: 4, y: 14 }, { x: 5, y: 1 }, { x: 6, y: 10 },
    { x: 7, y: 8 }, { x: 8, y: 6 }, { x: 9, y: 10 },
    { x: 10, y: 10 }, { x: 11, y: 16 }, { x: 12, y: 6 },
    { x: 13, y: 14 }, { x: 14, y: 7 }, { x: 15, y: 5 },
    { x: 16, y: 2 }, { x: 17, y: 14 }, { x: 18, y: 7 },
    { x: 19, y: 7 }, { x: 20, y: 10 }];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        valueType: 'Double',
        title: 'Overs',
        // Set the rangePadding as auto in Y Axis
        rangePadding: 'Auto'
    },
    primaryYAxis: {
        valueType: 'Double',
        title: 'Runs',
        // Set the rangePadding as auto in Y Axis
        rangePadding: 'Auto'
    },
    series:[{
        dataSource: chartData,
        xName: 'x', yName: 'y', width: 3,
        name: 'England', type: 'Line'
    }],
    title: 'England - Run Rate'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>

```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Label Format

Numeric Label Format

Numeric labels can be formatted by using the [labelFormat](#) property. Numeric labels supports all globalize format.

INDEX.JS

```

var chartData = [
    { x: 1900, y: 4, y1: 2.6, y2: 2.8 }, { x: 1920, y: 3.0, y1: 2.8, y2: 2.5 },
    { x: 1940, y: 3.8, y1: 2.6, y2: 2.8 }, { x: 1960, y: 3.4, y1: 3, y2: 3.2 },
    { x: 1980, y: 3.2, y1: 3.6, y2: 2.9 }, { x: 2000, y: 3.9, y1: 3, y2: 2 }
]
var chart= new ej.charts.Chart({
    primaryXAxis: {
        title: 'Year',
        edgeLabelPlacement: 'Shift'
    },
    primaryYAxis: {
        title: 'Sales Amount in Millions',
        //Label format as currency
        labelFormat: 'c'
    },
    series:[{
        dataSource: chartData, opacity: 0.6,
        xName: 'x', yName: 'y',
        name: 'Product X', type: 'Area'
    },{
        dataSource: chartData, opacity: 0.6,
        xName: 'x', yName: 'y1',
        name: 'Product Y', type: 'Area'
    },{
        dataSource: chartData, opacity: 0.6,
        xName: 'x', yName: 'y2',
        name: 'Product Z', type: 'Area'
    }],
    title: 'Average Sales Comparison'

```

```
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

The following table describes the result of applying some commonly used label formats on numeric values.

<!-- markdownlint-disable MD033 -->

Label Value	Label Format property value	Result	Description
1000	n1	1000.0	The Number is rounded to 1 decimal place
1000	n2	1000.00	The Number is rounded to 2 decimal place
1000	n3	1000.000	The Number is rounded to 3 decimal place
0.01	p1	1.0%	The Number is converted to percentage with 1 decimal place
0.01	p2	1.00%	The Number is converted to percentage with 2 decimal place
0.01	p3	1.000%	The Number is converted to percentage with 3 decimal place

1000	c1	\$1000.0	The Currency symbol is appended to number and number is rounded to 1 decimal place
1000	c2	\$1000.00	The Currency symbol is appended to number and number is rounded to 2 decimal place

GroupingSeparator

To separate groups of thousands, use [useGroupingSeparator](#) property in chart.

INDEX.JS

```
var chart = new ej.charts.Chart({
  series: [{
    dataSource: numericData,
    xName: 'x', yName: 'y',
    type: 'Column'
  }],
  useGroupingSeparator: true
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Custom Label Format

Axis also supports custom label format using placeholder like {value}°C, in which the value represent the axis label e.g 20°C.

INDEX.JS

```

var chartData = [
    { country: "USA", gold: 50, silver: 70, bronze: 45 },
    { country: "China", gold: 40, silver: 60, bronze: 55 },
    { country: "Japan", gold: 70, silver: 60, bronze: 50 },
    { country: "Australia", gold: 60, silver: 56, bronze: 40 },
    { country: "France", gold: 50, silver: 45, bronze: 35 },
    { country: "Germany", gold: 40, silver: 30, bronze: 22 },
    { country: "Italy", gold: 40, silver: 35, bronze: 37 },
    { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart= new ej.charts.Chart({
    primaryXAxis: {
        valueType: 'Category',
        title: 'Countries'
    },
    primaryYAxis: {
        minimum: 0, maximum: 80,
        interval: 20, title: 'Medals',
        // Custom label format
        labelFormat: '${value}K'
    },
    series:[{
        dataSource: chartData,
        xName: 'country', yName: 'gold',
        name: 'Gold', type: 'Column'
    }, {
        dataSource: chartData,
        xName: 'country', yName: 'silver',
        name: 'Silver', type: 'Column'
    }, {
        dataSource: chartData,
        xName: 'country', yName: 'bronze',
        name: 'Bronze', type: 'Column'
    }],
    title: 'Olympic Medals'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

```

```

    <div id="container">
      <div id="element"></div>
    </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

<!-- markdownlint-disable MD036 -->

Date time axis in ##Platform_Name## Chart control

DateTime Axis

Date time axis uses date time scale and displays the date time values as axis labels in the specified format.

INDEX.JS

```

var chartData = [
  { x: new Date(2000, 6, 11), y: 10 }, { x: new Date(2002, 3, 7), y: 30 },
  { x: new Date(2004, 3, 6), y: 15 }, { x: new Date(2006, 3, 30), y: 65 },
  { x: new Date(2008, 3, 8), y: 90 }, { x: new Date(2010, 3, 8), y: 85 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    // Date time scale in primary X Axis
    valueType: 'DateTime',
    title: 'Sales Across Years',
    labelFormat: 'yMMM'
  },
  primaryYAxis: {
    title: 'Sales Amount in millions(USD)'
  },
  series:[{
    dataSource: chartData,
    xName: 'x', yName: 'y',
    name: 'Sales', type: 'Line'
  }],
  title: 'Average Sales Comparison'
}, '#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Note: To use datetime axis, we need to inject DateTime using `Chart.Inject(DateTime)` method and set the [valueType](#) of axis to DateTime.

DateTimeCategory Axis

Date-time category axis is used to display the date-time values with non-linear intervals. For example, the business days alone have been depicted in a week here.

INDEX.JS

```

var chartData = [
    { x: new Date(2000, 6, 11), y: 10 }, { x: new Date(2002, 3, 7), y: 30 },
    { x: new Date(2004, 3, 6), y: 15 }, { x: new Date(2006, 3, 30), y: 65 },
    { x: new Date(2008, 3, 8), y: 90 }, { x: new Date(2010, 3, 8), y: 85 }
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        // Date time scale in primary X Axis
        valueType: 'DateTime',
        title: 'Sales Across Years',
        labelFormat: 'yMMM'
    },
    primaryYAxis: {
        title: 'Sales Amount in millions(USD)'
    },
    series:[{
        dataSource: chartData,
        xName: 'x', yName: 'y',
        name: 'Sales', type: 'Line'
    }],
    title: 'Average Sales Comparison'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>

```

```

<title>EJ2 Animation</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Note: To use `dateTimeCategory` axis, we need to inject `DateTimeCategory` using `Chart.Inject(DateTimeCategory)` method and set the [valueType](#) of axis to `DateTimeCategory`.

Range

Range of an axis will be calculated automatically based on the provided data, you can also customize the range of the axis using [minimum](#), [maximum](#) and [interval](#) property

of the axis.

INDEX.JS

```

var chartData = [
    { x: new Date(2000, 6, 11), y: 10 }, { x: new Date(2002, 3, 7), y: 30 },
    { x: new Date(2004, 3, 6), y: 15 }, { x: new Date(2006, 3, 30), y: 65 },
    { x: new Date(2008, 3, 8), y: 90 }, { x: new Date(2010, 3, 8), y: 85 }
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        // Date time range in primary X Axis
        valueType: 'DateTime',
        title: 'Sales Across Years',
        labelFormat: 'yMMM',
        minimum: new Date(2000, 6, 1),
        maximum: new Date(2010, 6, 1)
    },
    primaryYAxis: {
        title: 'Sales Amount in millions(USD)'
    },
    series:[{

```



```

        dataSource: chartData,
        xName: 'x', yName: 'y',
        name: 'Sales', type: 'Line'
    }],
    title: 'Average Sales Comparison'
}, '#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Interval Customization

Date time intervals can be customized by using the [interval](#) and [intervalType](#) properties of the axis. For example, when you set interval as 2 and intervalType as years, it considers 2 years as interval. Datetime axis supports following interval types,

- Auto
- Years
- Months
- Days
- Hours
- Minutes
- Seconds

INDEX.JS

```
var chartData = [
```

```

    { x: new Date(2000, 6, 11), y: 10 }, { x: new Date(2002, 3, 7), y: 30 },
    { x: new Date(2004, 3, 6), y: 15 }, { x: new Date(2006, 3, 30), y: 65 },
    { x: new Date(2008, 3, 8), y: 90 }, { x: new Date(2010, 3, 8), y: 85 }
  ];
  var chart = new ej.charts.Chart({
    primaryXAxis: {
      valueType: 'DateTime',
      title: 'Sales Across Years',
      labelFormat: 'yMMM',
      minimum: new Date(2000, 6, 1),
      maximum: new Date(2010, 6, 1),
      interval: 2,
      //interval type as years in primary x axis
      intervalType: 'Years'
    },
    primaryYAxis: {
      title: 'Sales Amount in millions(USD)'
    },
    series:[{
      dataSource: chartData,
      xName: 'x', yName: 'y',
      name: 'Sales', type: 'Line'
    }],
    title: 'Average Sales Comparison'
  }, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
  var ele = document.getElementById('container');
  if(ele) {
    ele.style.visibility = "visible";
  }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Applying Padding to the Range

Padding can be applied to the minimum and maximum extremes of the range by using the [rangePadding](#) property. Date time axis supports the following types of padding,

- None
- Round
- Additional

Datetime - None

When the [rangePadding](#) is set to **None**, minimum and maximum of an axis is based on the data.

INDEX.JS

```
var chartData = [
  { x: new Date(2000, 6, 11), y: 10 }, { x: new Date(2002, 3, 7), y: 30 },
  { x: new Date(2004, 3, 6), y: 15 }, { x: new Date(2006, 3, 30), y: 65 },
  { x: new Date(2008, 3, 8), y: 90 }, { x: new Date(2010, 3, 8), y: 85 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    valueType: 'DateTime',
    title: 'Sales Across Years',
    labelFormat: 'yMMM',
    //Range padding as none
    rangePadding: 'None'
  },
  primaryYAxis: {
    title: 'Sales Amount in millions(USD)'
  },
  series:[{
    dataSource: chartData,
    xName: 'x', yName: 'y',
    name: 'Sales', type: 'Line'
  }],
  title: 'Average Sales Comparison'
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>
```

```

<div id="container">
  <div id="element"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Datetime - Round

When the [rangePadding](#) is set to **Round**, minimum and maximum will be rounded to the nearest possible value, which is divisible by interval. For example, when the minimum is 15th Jan, interval is 1 and the interval type is 'month', then the axis minimum will be Jan 1st.

INDEX.JS

```

var chartData = [
  { x: new Date(2000, 6, 11), y: 10 }, { x: new Date(2002, 3, 7), y: 30 },
  { x: new Date(2004, 3, 6), y: 15 }, { x: new Date(2006, 3, 30), y: 65 },
  { x: new Date(2008, 3, 8), y: 90 }, { x: new Date(2010, 3, 8), y: 85 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    valueType: 'DateTime',
    title: 'Sales Across Years',
    labelFormat: 'yMMM',
    //Range padding as round
    rangePadding: 'Round'
  },
  primaryYAxis: {
    title: 'Sales Amount in millions(USD)'
  },
  series:[{
    dataSource: chartData,
    xName: 'x', yName: 'y',
    name: 'Sales', type: 'Line'
  }],
  title: 'Average Sales Comparison'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Datetime - Additional

When the [rangePadding](#) is set to **Additional**, interval of an axis will be padded to the minimum and maximum of the axis.

INDEX.JS

```

var chartData = [
    { x: new Date(2000, 6, 11), y: 10 }, { x: new Date(2002, 3, 7), y: 30 },
    { x: new Date(2004, 3, 6), y: 15 }, { x: new Date(2006, 3, 30), y: 65 },
    { x: new Date(2008, 3, 8), y: 90 }, { x: new Date(2010, 3, 8), y: 85 }
];
var chart= new ej.charts.Chart({
    primaryXAxis: {
        valueType: 'DateTime',
        title: 'Sales Across Years',
        labelFormat: 'yMMM',
        //Range padding as additional
        rangePadding: 'Additional'
    },
    primaryYAxis: {
        title: 'Sales Amount in millions(USD)'
    },
    series:[{
        dataSource: chartData,
        xName: 'x', yName: 'y',
        name: 'Sales', type: 'Line'
    }],
    title: 'Average Sales Comparison'
}, '#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

```

```

<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Label Format

You can format and parse the date to all globalize format using [labelFormat](#) property in an axis.

INDEX.JS

```

var chartData = [
    { x: new Date(2000, 6, 11), y: 10 }, { x: new Date(2002, 3, 7), y: 30 },
    { x: new Date(2004, 3, 6), y: 15 }, { x: new Date(2006, 3, 30), y: 65 },
    { x: new Date(2008, 3, 8), y: 90 }, { x: new Date(2010, 3, 8), y: 85 }
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        valueType: 'DateTime',
        title: 'Sales Across Years',
        //Label format as yMd
        labelFormat: 'yMd'
    },
    primaryYAxis: {
        title: 'Sales Amount in millions(USD)'
    },
    series:[{
        dataSource: chartData,
        xName: 'x', yName: 'y',
        name: 'Sales', type: 'Line'
    }],
    title: 'Average Sales Comparison'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>

```

```

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The following table describes the result of applying some common date time formats to the **labelFormat** property

<!-- markdownlint-disable MD033 -->

Label Value	Label Format Property Value	Result	Description
new Date(2000, 03, 10)	EEEE	Monday	The Date is displayed in day format
new Date(2000, 03, 10)	yMd	04/10/2000	The Date is displayed in month/date/year format
new Date(2000, 03, 10)	MMM	Apr	The Shorthand month for the date is displayed
new Date(2000, 03, 10)	hm	12:00 AM	Time of the date value is displayed as label
new Date(2000, 03, 10)	hms	12:00:00 AM	The Label is displayed in hours:minutes:seconds format

Custom Label Format

Axis also supports custom label format using placeholder like {value}°C, in which the value represent the axis label e.g 20°C.

INDEX.JS

```
var chartData = [
```

```

    { country: "USA", gold: 50, silver: 70, bronze: 45 },
    { country: "China", gold: 40, silver: 60, bronze: 55 },
    { country: "Japan", gold: 70, silver: 60, bronze: 50 },
    { country: "Australia", gold: 60, silver: 56, bronze: 40 },
    { country: "France", gold: 50, silver: 45, bronze: 35 },
    { country: "Germany", gold: 40, silver: 30, bronze: 22 },
    { country: "Italy", gold: 40, silver: 35, bronze: 37 },
    { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart= new ej.charts.Chart({
  primaryXAxis: {
    valueType: 'Category',
    title: 'Countries'
  },
  primaryYAxis: {
    minimum: 0, maximum: 80,
    interval: 20, title: 'Medals',
    // Custom label format
    labelFormat: '${value}K'
  },
  series:[{
    dataSource: chartData,
    xName: 'country', yName: 'gold',
    name: 'Gold', type: 'Column'
  }, {
    dataSource: chartData,
    xName: 'country', yName: 'silver',
    name: 'Silver', type: 'Column'
  }, {
    dataSource: chartData,
    xName: 'country', yName: 'bronze',
    name: 'Bronze', type: 'Column'
  }],
  title: 'Olympic Medals'
}, '#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
```



```

</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Logarithmic axis in ##Platform_Name## Chart control

<!-- markdownlint-disable MD033 -->

Logarithmic axis uses logarithmic scale and it is very useful in visualizing data, when it has numerical values in both lower order of magnitude (eg: 10⁻⁶) and higher order of magnitude (eg: 10⁶).

INDEX.JS

```

var chartData= [
    { x: new Date(1995, 0, 1), y: 80 }, { x: new Date(1996, 0, 1), y: 200 },
    { x: new Date(1997, 0, 1), y: 400 }, { x: new Date(1998, 0, 1), y: 600
},
    { x: new Date(1999, 0, 1), y: 700 }, { x: new Date(2000, 0, 1), y: 1400
},
    { x: new Date(2001, 0, 1), y: 2000 }, { x: new Date(2002, 0, 1), y: 4000
},
    { x: new Date(2003, 0, 1), y: 6000 }, { x: new Date(2004, 0, 1), y: 8000
},
    { x: new Date(2005, 0, 1), y: 11000 }
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        valueType: 'DateTime',
        title: 'Years',
        labelFormat: 'y'
    },
    primaryYAxis: {
        // Logarithmic scale in primary X Axis
        valueType: 'Logarithmic',
        title: 'Profit'
    },
    series:[{
        dataSource: chartData,
        xName: 'x', yName: 'y',
        name: 'Product X', type: 'Line'
    }],
    title: 'Product X Growth [1995-2005]'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
<title>EJ2 Animation</title>
<meta charset="utf-8">

```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Note: To use log axis, we need to inject **Logarithmic** using method **Chart.Inject(Logarithmic)** and set the **valueType** of axis to **Logarithmic**.

Range

Range of an axis, will be calculated automatically based on the provided data, you can also customize the range of an axis using **minimum**, **maximum** and **interval** property of the axis.

INDEX.JS

```

var chartData = [
    { x: new Date(1995, 0, 1), y: 80 }, { x: new Date(1996, 0, 1), y: 200 },
    { x: new Date(1997, 0, 1), y: 400 }, { x: new Date(1998, 0, 1), y: 600
},
    { x: new Date(1999, 0, 1), y: 700 }, { x: new Date(2000, 0, 1), y: 1400
},
    { x: new Date(2001, 0, 1), y: 2000 }, { x: new Date(2002, 0, 1), y: 4000
},
    { x: new Date(2003, 0, 1), y: 6000 }, { x: new Date(2004, 0, 1), y: 8000
},
    { x: new Date(2005, 0, 1), y: 11000 }
];
var chart= new ej.charts.Chart({
    primaryXAxis: {
        valueType: 'DateTime',
        title: 'Years',
        labelFormat: 'y'
    },
    primaryYAxis: {
        //Logarithmic scale range in primary X Axis
        valueType: 'Logarithmic',
        title: 'Profit',

```

```

        minimum: 100,
        maximum: 10000
    },
    series:[{
        dataSource: chartData,
        xName: 'x', yName: 'y',
        name: 'Product X', type: 'Line'
    }],
    title: 'Product X Growth [1995-2005]'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Logarithmic Base

Logarithmic base can be customized by using the [logBase](#) property of the axis. For example when the logBase is 5, the axis values follows 5^{-2} , 5^{-1} , 5^0 , 5^1 , 5^2 etc.

INDEX.JS

```

var chartData= [
  { x: new Date(1995, 0, 1), y: 80 }, { x: new Date(1996, 0, 1), y: 200 },
  { x: new Date(1997, 0, 1), y: 400 }, { x: new Date(1998, 0, 1), y: 600
},
  { x: new Date(1999, 0, 1), y: 700 }, { x: new Date(2000, 0, 1), y: 1400
},

```

```

    { x: new Date(2001, 0, 1), y: 2000 }, { x: new Date(2002, 0, 1), y: 4000
  },
  { x: new Date(2003, 0, 1), y: 6000 }, { x: new Date(2004, 0, 1), y: 8000
  },
  { x: new Date(2005, 0, 1), y: 11000 }
];
let chart= new ej.charts.Chart({
  primaryXAxis: {
    valueType: 'DateTime',
    title: 'Years',
    labelFormat: 'y'
  },
  primaryYAxis: {
    valueType: 'Logarithmic',
    title: 'Profit',
    // logBase for logarithmic scale
    logBase: 2
  },
  series:[{
    dataSource: chartData,
    xName: 'x', yName: 'y',
    name: 'Product X', type: 'Line'
  }],
  title: 'Product X Growth [1995-2005]'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Logarithmic Interval

Logarithmic axis interval can be customized by using the [interval](#) property of the axis. When the logarithmic base is 10 and logarithmic interval is 2, then the axis labels are placed at an interval of 10^2 . The default value of the interval is 1.

INDEX.JS

```
var chartData = [
  { x: new Date(1995, 0, 1), y: 80 }, { x: new Date(1996, 0, 1), y: 200 },
  { x: new Date(1997, 0, 1), y: 400 }, { x: new Date(1998, 0, 1), y: 600 },
  { x: new Date(1999, 0, 1), y: 700 }, { x: new Date(2000, 0, 1), y: 1400 },
  { x: new Date(2001, 0, 1), y: 2000 }, { x: new Date(2002, 0, 1), y: 4000 },
  { x: new Date(2003, 0, 1), y: 6000 }, { x: new Date(2004, 0, 1), y: 8000 },
  { x: new Date(2005, 0, 1), y: 11000 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    valueType: 'DateTime',
    title: 'Years',
    labelFormat: 'y'
  },
  primaryYAxis: {
    //Logarithmic interval in primary X Axis
    valueType: 'Logarithmic',
    title: 'Profit',
    interval: 2
  },
  series:[{
    dataSource: chartData,
    xName: 'x', yName: 'y',
    name: 'Product X', type: 'Line'
  }],
  title: 'Product X Growth [1995-2005]'
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>
```

```

    <div id="container">
      <div id="element"></div>
    </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Axis labels in ##Platform_Name## Chart control

Smart Axis Labels

When the axis labels overlap with each other, you can use [labelIntersectAction](#) property in the axis, to place them smartly.

When setting `labelIntersectAction` as `Hide`

INDEX.JS

```

var chartData = [
  { x: "South Korea", y: 39.4 }, { x: "India", y: 61.3 }, { x: "Pakistan", y: 20.4 },
  { x: "Germany", y: 65.1 }, { x: "Australia", y: 15.8 }, { x: "Italy", y: 29.2 },
  { x: "United Kingdom", y: 44.6 }, { x: "Saudi Arabia", y: 9.7 }, { x: "Russia", y: 40.8 },
  { x: "Mexico", y: 31 }, { x: "Brazil", y: 75.9 }, { x: "China", y: 51.4 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    title: 'Countries',
    valueType: 'Category',
    //label intersect as hide
    labelIntersectAction: 'Hide'
  },
  primaryYAxis: {
    minimum: 0, maximum: 80, interval: 10,
    title: 'People(in millions)'
  },
  axes: [
    {
      majorGridLines: { width: 0 },
      rowIndex: 0, opposedPosition: true,
      lineStyle: { width: 0 },
      minimum: 24, maximum: 36, interval: 2,
      name: 'yAxis', title: 'Temperature (Celsius)',
      labelFormat: '{value}°C'
    }
  ],
  series: [{
    dataSource: chartData,
    xName: 'x', yName: 'y',

```

```

        name: 'Internet', type: 'Column'
    }],
    title: 'Internet Users'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

When setting `labelIntersectAction` as `Rotate45`

INDEX.JS

```

var chartData = [
  { x: "South Korea", y: 39.4 }, { x: "India", y: 61.3 }, { x: "Pakistan",
y: 20.4 },
  { x: "Germany", y: 65.1 }, { x: "Australia", y: 15.8 }, { x: "Italy", y:
29.2 },
  { x: "United Kingdom", y: 44.6 }, { x: "Saudi Arabia", y: 9.7 }, { x:
"Russia", y: 40.8 },
  { x: "Mexico", y: 31 }, { x: "Brazil", y: 75.9 }, { x: "China", y: 51.4
}
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    title: 'Countries',
    valueType: 'Category',
    // label intersect as 45
    labelIntersectAction: 'Rotate45'
  },

```

```

primaryYAxis: {
    minimum: 0, maximum: 80, interval: 10,
    title: 'People(in millions)'
},
axes:[
    {
        majorGridLines: { width: 0 },
        rowIndex: 0, opposedPosition: true,
        lineStyle: { width: 0 },
        minimum: 24, maximum: 36, interval: 2,
        name: 'yAxis', title: 'Temperature (Celsius)',
        labelFormat: '{value}°C'
    }
],
series:[{
    dataSource: chartData,
    xName: 'x', yName: 'y',
    name: 'Internet', type: 'Column'
}],
title: 'Internet Users'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

When setting `labelIntersectAction` as `Rotate90`

INDEX.JS


```

var chartData = [
    { x: "South Korea", y: 39.4 }, { x: "India", y: 61.3 }, { x: "Pakistan",
y: 20.4 },
    { x: "Germany", y: 65.1 }, { x: "Australia", y: 15.8 }, { x: "Italy", y:
29.2 },
    { x: "United Kingdom", y: 44.6 }, { x: "Saudi Arabia", y: 9.7 }, { x:
"Russia", y: 40.8 },
    { x: "Mexico", y: 31 }, { x: "Brazil", y: 75.9 }, { x: "China", y: 51.4
}
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        title: 'Countries',
        valueType: 'Category',
        // label intersect as 90
        labelIntersectAction: 'Rotate90'
    },
    primaryYAxis: {
        minimum: 0, maximum: 80, interval: 10,
        title: 'People(in millions)'
    },
    axes:[
        {
            majorGridLines: { width: 0 },
            rowIndex: 0, opposedPosition: true,
            lineStyle: { width: 0 },
            minimum: 24, maximum: 36, interval: 2,
            name: 'yAxis', title: 'Temperature (Celsius)',
            labelFormat: '{value}°C'
        }
    ],
    series:[{
        dataSource: chartData,
        xName: 'x', yName: 'y',
        name: 'Internet', type: 'Column'
    }],
    title: 'Internet Users'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```

```

    <div id="container">
      <div id="element"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Axis Labels Positioning

By default, the axis labels can be placed at **outside** the axis line and this also can be placed at **inside** the axis line using the **labelPosition** property.

INDEX.JS

```

var chartData = [
  { country: "USA", gold: 50, silver: 70, bronze: 45 },
  { country: "China", gold: 40, silver: 60, bronze: 55 },
  { country: "Japan", gold: 70, silver: 60, bronze: 50 },
  { country: "Australia", gold: 60, silver: 56, bronze: 40 },
  { country: "France", gold: 50, silver: 45, bronze: 35 },
  { country: "Germany", gold: 40, silver: 30, bronze: 22 },
  { country: "Italy", gold: 40, silver: 35, bronze: 37 },
  { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    valueType: 'Category',
    // label placement as on ticks
    labelPlacement: 'OnTicks',
    title: 'Countries'
  },
  primaryYAxis: {
    minimum: 0, maximum: 80,
    interval: 20, title: 'Medals'
  },
  series:[{
    dataSource: chartData,
    xName: 'country', yName: 'gold',
    name: 'Gold', type: 'Column'
  }, {
    dataSource: chartData,
    xName: 'country', yName: 'silver',
    name: 'Silver', type: 'Column'
  }, {
    dataSource: chartData,
    xName: 'country', yName: 'bronze',
    name: 'Bronze', type: 'Column'
  }],
  title: 'Olympic Medals'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Multilevel Labels

Any number of levels of labels can be added to an axis using the `multiLevelLabels` property. This property can be configured using the following properties:

- Categories
- Overflow
- Alignment
- Text style
- Border

Note: To use multilevel label feature, we need to inject `MultiLevelLabel` using `Chart.Inject(MultiLevelLabel)` method.

Categories

Using the categories property, you can configure the `start`, `end`, `text`, and `maximumTextWidth` of multilevel labels.

INDEX.JS

```

var chartData = [
  { country: "USA", gold: 50, silver: 70, bronze: 45 },
  { country: "China", gold: 40, silver: 60, bronze: 55 },
  { country: "Japan", gold: 70, silver: 60, bronze: 50 },

```

```

    { country: "Australia", gold: 60, silver: 56, bronze: 40 },
    { country: "France", gold: 50, silver: 45, bronze: 35 },
    { country: "Germany", gold: 40, silver: 30, bronze: 22 },
    { country: "Italy", gold: 40, silver: 35, bronze: 37 },
    { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
  ];
  var chart = new ej.charts.Chart({
    primaryXAxis: {
      valueType: 'Category',
      // label placement as on ticks
      labelPlacement: 'OnTicks',
      title: 'Countries'
    },
    primaryYAxis: {
      minimum: 0, maximum: 80,
      interval: 20, title: 'Medals'
    },
    series:[{
      dataSource: chartData,
      xName: 'country', yName: 'gold',
      name: 'Gold', type: 'Column'
    }, {
      dataSource: chartData,
      xName: 'country', yName: 'silver',
      name: 'Silver', type: 'Column'
    }, {
      dataSource: chartData,
      xName: 'country', yName: 'bronze',
      name: 'Bronze', type: 'Column'
    }],
    title: 'Olympic Medals'
  }, '#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
  var ele = document.getElementById('container');
```

```

if(ele) {
    ele.style.visibility = "visible";
}

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Overflow

Using the `overflow` property, you can `trim` or `wrap` the multilevel labels.

INDEX.JS

```

var chartData = [
    { country: "USA", gold: 50, silver: 70, bronze: 45 },
    { country: "China", gold: 40, silver: 60, bronze: 55 },
    { country: "Japan", gold: 70, silver: 60, bronze: 50 },
    { country: "Australia", gold: 60, silver: 56, bronze: 40 },
    { country: "France", gold: 50, silver: 45, bronze: 35 },
    { country: "Germany", gold: 40, silver: 30, bronze: 22 },
    { country: "Italy", gold: 40, silver: 35, bronze: 37 },
    { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        valueType: 'Category',
        // label placement as on ticks
        labelPlacement: 'OnTicks',
        title: 'Countries'
    },
    primaryYAxis: {
        minimum: 0, maximum: 80,
        interval: 20, title: 'Medals'
    },
    series:[{
        dataSource: chartData,
        xName: 'country', yName: 'gold',
        name: 'Gold', type: 'Column'
    }, {
        dataSource: chartData,
        xName: 'country', yName: 'silver',
        name: 'Silver', type: 'Column'
    }, {
        dataSource: chartData,
        xName: 'country', yName: 'bronze',
        name: 'Bronze', type: 'Column'
    }],
    title: 'Olympic Medals'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">

```

```

<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Alignment

The **alignment** property provides option to position the multilevel labels at **far**, **center**, or **near**.

INDEX.JS

```

var chartData = [
    { country: "USA", gold: 50, silver: 70, bronze: 45 },
    { country: "China", gold: 40, silver: 60, bronze: 55 },
    { country: "Japan", gold: 70, silver: 60, bronze: 50 },
    { country: "Australia", gold: 60, silver: 56, bronze: 40 },
    { country: "France", gold: 50, silver: 45, bronze: 35 },
    { country: "Germany", gold: 40, silver: 30, bronze: 22 },
    { country: "Italy", gold: 40, silver: 35, bronze: 37 },
    { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        valueType: 'Category',
        // label placement as on ticks
        labelPlacement: 'OnTicks',
        title: 'Countries'
    },
    primaryYAxis: {
        minimum: 0, maximum: 80,
        interval: 20, title: 'Medals'
    },
    series:[{
        dataSource: chartData,
        xName: 'country', yName: 'gold',
        name: 'Gold', type: 'Column'
    }, {
        dataSource: chartData,
        xName: 'country', yName: 'silver',

```

```

        name: 'Silver', type: 'Column'
    }, {
        dataSource: chartData,
        xName: 'country', yName: 'bronze',
        name: 'Bronze', type: 'Column'
    }],
    title: 'Olympic Medals'
}, '#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Text customization

The `textStyle` property of multilevel labels provides options to customize the `size`, `color`, `fontFamily`, `fontWeight`, `fontStyle`, `opacity`, `textAlignment` and `textOverflow`.

INDEX.JS

```

var chartData = [
  { country: "USA", gold: 50, silver: 70, bronze: 45 },
  { country: "China", gold: 40, silver: 60, bronze: 55 },
  { country: "Japan", gold: 70, silver: 60, bronze: 50 },
  { country: "Australia", gold: 60, silver: 56, bronze: 40 },
  { country: "France", gold: 50, silver: 45, bronze: 35 },
  { country: "Germany", gold: 40, silver: 30, bronze: 22 },
  { country: "Italy", gold: 40, silver: 35, bronze: 37 },
  { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
```

```

var chart = new ej.charts.Chart({
  primaryXAxis: {
    valueType: 'Category',
    // label placement as on ticks
    labelPlacement: 'OnTicks',
    title: 'Countries'
  },
  primaryYAxis: {
    minimum: 0, maximum: 80,
    interval: 20, title: 'Medals'
  },
  series:[{
    dataSource: chartData,
    xName: 'country', yName: 'gold',
    name: 'Gold', type: 'Column'
  }, {
    dataSource: chartData,
    xName: 'country', yName: 'silver',
    name: 'Silver', type: 'Column'
  }, {
    dataSource: chartData,
    xName: 'country', yName: 'bronze',
    name: 'Bronze', type: 'Column'
  }],
  title: 'Olympic Medals'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```


Border customization

Using the `border` property, you can customize the width, color, and type. The type of border are `Rectangle`, `Brace`, `WithoutBorder`, `WithoutTopBorder`, `WithoutTopandBottomBorder` and `CurlyBrace`.

INDEX.JS

```
var chartData = [
  { country: "USA", gold: 50, silver: 70, bronze: 45 },
  { country: "China", gold: 40, silver: 60, bronze: 55 },
  { country: "Japan", gold: 70, silver: 60, bronze: 50 },
  { country: "Australia", gold: 60, silver: 56, bronze: 40 },
  { country: "France", gold: 50, silver: 45, bronze: 35 },
  { country: "Germany", gold: 40, silver: 30, bronze: 22 },
  { country: "Italy", gold: 40, silver: 35, bronze: 37 },
  { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    valueType: 'Category',
    // label placement as on ticks
    labelPlacement: 'OnTicks',
    title: 'Countries'
  },
  primaryYAxis: {
    minimum: 0, maximum: 80,
    interval: 20, title: 'Medals'
  },
  series:[{
    dataSource: chartData,
    xName: 'country', yName: 'gold',
    name: 'Gold', type: 'Column'
  }, {
    dataSource: chartData,
    xName: 'country', yName: 'silver',
    name: 'Silver', type: 'Column'
  }, {
    dataSource: chartData,
    xName: 'country', yName: 'bronze',
    name: 'Bronze', type: 'Column'
  }],
  title: 'Olympic Medals'
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Edge Label Placement

Labels with long text at the edges of an axis may appear partially in the chart. To avoid this, use [edgeLabelPlacement](#) property in axis, which moves the label inside the chart area for better appearance or hides it.

INDEX.JS

```

var chartData = [
    { country: "USA", gold: 50, silver: 70, bronze: 45 },
    { country: "China", gold: 40, silver: 60, bronze: 55 },
    { country: "Japan", gold: 70, silver: 60, bronze: 50 },
    { country: "Australia", gold: 60, silver: 56, bronze: 40 },
    { country: "France", gold: 50, silver: 45, bronze: 35 },
    { country: "Germany", gold: 40, silver: 30, bronze: 22 },
    { country: "Italy", gold: 40, silver: 35, bronze: 37 },
    { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        valueType: 'Category',
        // label placement as on ticks
        labelPlacement: 'OnTicks',
        title: 'Countries'
    },
    primaryYAxis: {
        minimum: 0, maximum: 80,
        interval: 20, title: 'Medals'
    },
    series:[{
        dataSource: chartData,
        xName: 'country', yName: 'gold',
        name: 'Gold', type: 'Column'
    }, {
        dataSource: chartData,
        xName: 'country', yName: 'silver',

```

```

        name: 'Silver', type: 'Column'
    }, {
        dataSource: chartData,
        xName: 'country', yName: 'bronze',
        name: 'Bronze', type: 'Column'
    }],
    title: 'Olympic Medals'
}, '#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Sorting

The chart's data source can be sorted using the `sort` method of chart. The arguments that are required to pass to sort method are data of chart. The fields depend on which sorting is performed either `x` or `y`, and the `isDescending` with which data source values are sorted in either `ascending` or `descending`.

INDEX.JS

```

var chartData = [
  { country: "USA", gold: 50, silver: 70, bronze: 45 },
  { country: "China", gold: 40, silver: 60, bronze: 55 },
  { country: "Japan", gold: 70, silver: 60, bronze: 50 },
  { country: "Australia", gold: 60, silver: 56, bronze: 40 },
  { country: "France", gold: 50, silver: 45, bronze: 35 },
  { country: "Germany", gold: 40, silver: 30, bronze: 22 },
  { country: "Italy", gold: 40, silver: 35, bronze: 37 },
  { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
]
```

```

];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    valueType: 'Category',
    // label placement as on ticks
    labelPlacement: 'OnTicks',
    title: 'Countries'
  },
  primaryYAxis: {
    minimum: 0, maximum: 80,
    interval: 20, title: 'Medals'
  },
  series:[{
    dataSource: chartData,
    xName: 'country', yName: 'gold',
    name: 'Gold', type: 'Column'
  }, {
    dataSource: chartData,
    xName: 'country', yName: 'silver',
    name: 'Silver', type: 'Column'
  }, {
    dataSource: chartData,
    xName: 'country', yName: 'bronze',
    name: 'Bronze', type: 'Column'
  }],
  title: 'Olympic Medals'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
<script src="index.js" type="text/javascript"></script>

```

```
</body></html>
```

Labels Customization

Border of the axis labels can be customized using `width`, `color` and `typr` property of the axis.

INDEX.JS

```
var chartData = [
  { country: "USA", gold: 50, silver: 70, bronze: 45 },
  { country: "China", gold: 40, silver: 60, bronze: 55 },
  { country: "Japan", gold: 70, silver: 60, bronze: 50 },
  { country: "Australia", gold: 60, silver: 56, bronze: 40 },
  { country: "France", gold: 50, silver: 45, bronze: 35 },
  { country: "Germany", gold: 40, silver: 30, bronze: 22 },
  { country: "Italy", gold: 40, silver: 35, bronze: 37 },
  { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    valueType: 'Category',
    // label placement as on ticks
    labelPlacement: 'OnTicks',
    title: 'Countries'
  },
  primaryYAxis: {
    minimum: 0, maximum: 80,
    interval: 20, title: 'Medals'
  },
  series:[{
    dataSource: chartData,
    xName: 'country', yName: 'gold',
    name: 'Gold', type: 'Column'
  }, {
    dataSource: chartData,
    xName: 'country', yName: 'silver',
    name: 'Silver', type: 'Column'
  }, {
    dataSource: chartData,
    xName: 'country', yName: 'bronze',
    name: 'Bronze', type: 'Column'
  }],
  title: 'Olympic Medals'
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customizing Specific Point

You can customize the specific text in the axis labels using `axisLabelRender` event.

INDEX.JS

```

var chartData = [
    { country: "USA", gold: 50, silver: 70, bronze: 45 },
    { country: "China", gold: 40, silver: 60, bronze: 55 },
    { country: "Japan", gold: 70, silver: 60, bronze: 50 },
    { country: "Australia", gold: 60, silver: 56, bronze: 40 },
    { country: "France", gold: 50, silver: 45, bronze: 35 },
    { country: "Germany", gold: 40, silver: 30, bronze: 22 },
    { country: "Italy", gold: 40, silver: 35, bronze: 37 },
    { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        valueType: 'Category',
        // label placement as on ticks
        labelPlacement: 'OnTicks',
        title: 'Countries'
    },
    primaryYAxis: {
        minimum: 0, maximum: 80,
        interval: 20, title: 'Medals'
    },
    series:[{
        dataSource: chartData,
        xName: 'country', yName: 'gold',
        name: 'Gold', type: 'Column'
    }, {
        dataSource: chartData,
        xName: 'country', yName: 'silver',
        name: 'Silver', type: 'Column'
    }, {
        dataSource: chartData,
        xName: 'country', yName: 'bronze',

```

```

        name: 'Bronze', type: 'Column'
    }],
    title: 'Olympic Medals'
}, '#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Line break support

Line break feature used to customize the long axis label text into multiple lines by using tag. Refer the below example in that dataSource x value contains long text, it breaks into two lines by using `
` tag.

INDEX.JS

```

var chart = new ej.charts.Chart({
  //Initializing Primary X Axis
  primaryXAxis: {
    title: 'Country',
    valueType: 'Category',
    majorGridLines: { width: 0 },
    enableTrim: false
  },
  //Initializing Primary Y Axis
  primaryYAxis: {
    minimum: 0,
    maximum: 800,
    labelFormat: '{value}M',
  },
});
```

```

series: [
  {
    type: 'Bar', tooltipMappingName: 'country',
    dataSource: [
      { x: 'Germany', y: 72, country: 'GER: 72' },
      { x: 'Russia', y: 103.1, country: 'RUS: 103.1' },
      { x: 'Brazil', y: 139.1, country: 'BRZ: 139.1' },
      { x: 'India', y: 462.1, country: 'IND: 462.1' },
      { x: 'China', y: 721.4, country: 'CHN: 721.4' },
      { x: 'United States <br> Of America', y: 286.9, country:
'USA: 286.9' },
      { x: 'Great Britain', y: 115.1, country: 'GBR: 115.1' },
      { x: 'Nigeria', y: 97.2, country: 'NGR: 97.2' },
    ],
    xName: 'x', width: 2,
    yName: 'y', marker: {
      dataLabel: {
        visible: true,
        position: 'Top', font: {
          fontWeight: '600',
          color: '#ffffff'
        }
      }
    },
    name: 'Users'
  },
],
legendSettings: {
  visible: false
},
//Initializing Chart Title
title: 'Internet Users - 2016',
//Initializing Tooltip
tooltip: { enable: true, format: '${point.tooltip}' }
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>

```



```

</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Maximum Labels

MaximumLabels property is set, then the labels will be rendered based on the count in the property per 100 pixel. If you have set range (minimum, maximum, interval) and maximumLabels, then the priority goes to range only. If you haven't set the range, then we have considered priority to maximumLabels property.

INDEX.JS

```

var chart = new ej.charts.Chart({
    //Initializing Primary X Axis
    primaryXAxis: {
        title: 'Country',
        valueType: 'Category',
        majorGridLines: { width: 0 },
        enableTrim: false,
        maximumLabels: 1
    },
    //Initializing Primary Y Axis
    primaryYAxis: {
        minimum: 0,
        maximum: 800,
        labelFormat: '{value}M',
    },
    series: [
        {
            type: 'Bar', tooltipMappingName: 'country',
            dataSource: [
                { x: 'Germany', y: 72, country: 'GER: 72' },
                { x: 'Russia', y: 103.1, country: 'RUS: 103.1' },
                { x: 'Brazil', y: 139.1, country: 'BRZ: 139.1' },
                { x: 'India', y: 462.1, country: 'IND: 462.1' },
                { x: 'China', y: 721.4, country: 'CHN: 721.4' },
                { x: 'United States <br> Of America', y: 286.9, country:
'USA: 286.9' },
                { x: 'Great Britain', y: 115.1, country: 'GBR: 115.1' },
                { x: 'Nigeria', y: 97.2, country: 'NGR: 97.2' },
            ],
            xName: 'x', width: 2,
            yName: 'y', marker: {
                dataLabel: {
                    visible: true,
                    position: 'Top', font: {
                        fontWeight: '600',
                        color: '#ffffff'
                    }
                }
            }
        }
    ]
});

```

```

        },
        name: 'Users'
    },
    ],
    legendSettings: {
        visible: false
    },
    //Initializing Chart Title
    title: 'Internet Users - 2016',
    //Initializing Tooltip
    tooltip: { enable: true, format: '${point.tooltip}' }
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

Axis customization in ##Platform_Name## Chart control

Axis Crossing

An axis can be positioned in the chart area using `crossesAt` and `crossesInAxis` properties. The `crossesAt` property specifies the values (datetime, numeric, or logarithmic) at which the axis line has to be intersected with the vertical axis or vice-versa, and the `crossesInAxis` property specifies the axis name with which the axis line has to be crossed.

INDEX.JS

```

var chartData = [

```

```

    { country: "USA", gold: 50, silver: 70, bronze: 45 },
    { country: "China", gold: 40, silver: 60, bronze: 55 },
    { country: "Japan", gold: 70, silver: 60, bronze: 50 },
    { country: "Australia", gold: 60, silver: 56, bronze: 40 },
    { country: "France", gold: 50, silver: 45, bronze: 35 },
    { country: "Germany", gold: 40, silver: 30, bronze: 22 },
    { country: "Italy", gold: 40, silver: 35, bronze: 37 },
    { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    valueType: 'Category',
    // label placement as on ticks
    labelPlacement: 'OnTicks',
    title: 'Countries'
  },
  primaryYAxis: {
    minimum: 0, maximum: 80,
    interval: 20, title: 'Medals'
  },
  series:[{
    dataSource: chartData,
    xName: 'country', yName: 'gold',
    name: 'Gold', type: 'Column'
  }, {
    dataSource: chartData,
    xName: 'country', yName: 'silver',
    name: 'Silver', type: 'Column'
  }, {
    dataSource: chartData,
    xName: 'country', yName: 'bronze',
    name: 'Bronze', type: 'Column'
  }],
  title: 'Olympic Medals'
}, '#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
```

```

    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Title

You can add a title to the axis using [title](#) property to provide quick information to the user about the data plotted in the axis. Title style can be customized using `titleStyle` property of the axis.

INDEX.JS

```

var chartData = [
    { country: "USA", gold: 50, silver: 70, bronze: 45 },
    { country: "China", gold: 40, silver: 60, bronze: 55 },
    { country: "Japan", gold: 70, silver: 60, bronze: 50 },
    { country: "Australia", gold: 60, silver: 56, bronze: 40 },
    { country: "France", gold: 50, silver: 45, bronze: 35 },
    { country: "Germany", gold: 40, silver: 30, bronze: 22 },
    { country: "Italy", gold: 40, silver: 35, bronze: 37 },
    { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        valueType: 'Category',
        title: 'Countries'
    },
    primaryYAxis: {
        minimum: 0, maximum: 80,
        interval: 20, title: 'Medals',
        labelFormat: '${value}K',
        //Axis title text style
        titleStyle: {
            size: '16px', color: 'grey',
            fontFamily: 'Segoe UI', fontWeight: 'bold'
        }
    },
    series:[{
        dataSource: chartData,
        xName: 'country', yName: 'gold',
        name: 'Gold', type: 'Column'
    }, {
        dataSource: chartData,
        xName: 'country', yName: 'silver',
        name: 'Silver', type: 'Column'
    }, {
        dataSource: chartData,
        xName: 'country', yName: 'bronze',
        name: 'Bronze', type: 'Column'
    }],
    title: 'Olympic Medals'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Title Rotation

By using the [titleRotation](#) property, you can rotate the axis title from 0 to 360 degree.

INDEX.JS

```

var chartData = [
  { country: "USA", gold: 50, silver: 70, bronze: 45 },
  { country: "China", gold: 40, silver: 60, bronze: 55 },
  { country: "Japan", gold: 70, silver: 60, bronze: 50 },
  { country: "Australia", gold: 60, silver: 56, bronze: 40 },
  { country: "France", gold: 50, silver: 45, bronze: 35 },
  { country: "Germany", gold: 40, silver: 30, bronze: 22 },
  { country: "Italy", gold: 40, silver: 35, bronze: 37 },
  { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    valueType: 'Category',
    title: 'Countries', titleRotation: 90
  },
  primaryYAxis: {
    minimum: 0, maximum: 80,
    interval: 20, title: 'Medals',
    labelFormat: '${value}K',
    //Axis title text style

```

```

        titleStyle: {
            size: '16px', color: 'grey',
            fontFamily : 'Segoe UI', fontWeight : 'bold'
        },
    },
    series:[{
        dataSource: chartData,
        xName: 'country', yName: 'gold',
        name: 'Gold', type: 'Column'
    }, {
        dataSource: chartData,
        xName: 'country', yName: 'silver',
        name: 'Silver', type: 'Column'
    }, {
        dataSource: chartData,
        xName: 'country', yName: 'bronze',
        name: 'Bronze', type: 'Column'
    }],
    title: 'Olympic Medals'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Tick Lines Customization

You can customize the **width**, **color** and **size** of the minor and major tick lines, using [majorTickLines](#) and

[minorTickLines](#) properties in the axis.

INDEX.JS

```
var chartData= [
  { x: 'Jan', y: 60 }, { x: 'Feb', y: 50 }, { x: 'Mar', y: 64 },
  { x: 'Apr', y: 63 }, { x: 'May', y: 81 }, { x: 'Jun', y: 64 },
  { x: 'Jul', y: 82 }, { x: 'Aug', y: 96 }, { x: 'Sep', y: 78 },
  { x: 'Oct', y: 60 }, { x: 'Nov', y: 58 }, { x: 'Dec', y: 56 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    valueType: 'Category',
    //Tick lines customization
    majorTickLines : {
      color : 'blue',
      width : 5
    },
    minorTickLines : {
      color : 'red',
      width : 0
    }
  },
  primaryYAxis: {
    title: 'Temperature (Fahrenheit)',
    //Grid lines customization
    majorTickLines : {
      color : 'blue',
      width : 5
    },
    minorTickLines : {
      color : 'red',
      width : 0
    }
  },
  series:[{
    dataSource: chartData,
    xName: 'x', yName: 'y',
    name: 'Sales', type: 'Column'
  }],
  title: 'Temperature flow over months'
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Grid Lines Customization

You can customize the **width**, **color** and **dashArray** of the minor and major grid lines, using [majorGridLines](#) and [minorGridLines](#) properties in the axis.

INDEX.JS

```

var chartData= [
    { x: 'Jan', y: 60 }, { x: 'Feb', y: 50 }, { x: 'Mar', y: 64 },
    { x: 'Apr', y: 63 }, { x: 'May', y: 81 }, { x: 'Jun', y: 64 },
    { x: 'Jul', y: 82 }, { x: 'Aug', y: 96 }, { x: 'Sep', y: 78 },
    { x: 'Oct', y: 60 }, { x: 'Nov', y: 58 }, { x: 'Dec', y: 56 }
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        valueType: 'Category',
        //Grid lines customization
        majorGridLines : {
            color : 'blue',
            width : 1
        },
        minorGridLines : {
            color : 'red',
            width : 0
        }
    },
    primaryYAxis: {
        title: 'Temperature (Fahrenheit)',
        //Grid lines customization
        majorGridLines : {
            color : 'blue',
            width : 1
        },
        minorGridLines : {
            color : 'red',
            width : 0
        }
    },
});

```



```

series:[{
  dataSource: chartData,
  xName: 'x', yName: 'y',
  name: 'Sales', type: 'Column'
}],
title: 'Temperature flow over months'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Multiple Axis

In addition to primary X and Y axis, we can add n number of axis to the chart. Series can be associated with this axis, by mapping with axis's unique name.

INDEX.JS

```

var chartData = [
  { x: 'Jan', y: 15, y1: 33 }, { x: 'Feb', y: 20, y1: 31 }, { x: 'Mar', y:
  35, y1: 30 },
  { x: 'Apr', y: 40, y1: 28 }, { x: 'May', y: 80, y1: 29 }, { x: 'Jun', y:
  70, y1: 30 },
  { x: 'Jul', y: 65, y1: 33 }, { x: 'Aug', y: 55, y1: 32 }, { x: 'Sep', y:
  50, y1: 34 },
  { x: 'Oct', y: 30, y1: 32 }, { x: 'Nov', y: 35, y1: 32 }, { x: 'Dec', y:
  35, y1: 31 }
];
var chart = new ej.charts.Chart({

```

```

primaryXAxis: {
    title: 'Months',
    valueType: 'Category',
    interval: 1
},
primaryYAxis: {
    minimum: 0, maximum: 90, interval: 10,
    lineStyle: { width: 0 },
    title: 'Temperature (Fahrenheit)',
    labelFormat: '{value}°F'
},
// Initializing multiple axis
axes:[
    {
        majorGridLines: { width: 0 },
        rowIndex: 0, opposedPosition: true,
        lineStyle: { width: 0 },
        minimum: 24, maximum: 36, interval: 2,
        name: 'yAxis', title: 'Temperature (Celsius)',
        labelFormat: '{value}°C'
    }
],
series:[{
    dataSource: chartData,
    xName: 'x', yName: 'y',
    name: 'Germany', type: 'Column'
},{
    dataSource: chartData, width:2,
    xName: 'x', yName: 'y1', yAxisName: 'yAxis',
    name: 'Japan', type: 'Line',
    marker: { visible: true, width: 10, height: 10, border: { width: 2,
color: '#F8AB1D' } }
    },
    title: 'Weather Condition'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>

```

```

    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Inversed Axis

<!-- markdownlint-disable MD033 -->

When an axis is inversed, highest value of the axis comes closer to origin and vice versa. To place an axis in inversed manner set this property

[isInversed](#) to true.

INDEX.JS

```

var chart = new ej.charts.Chart({
    //Initializing Primary X Axis
    primaryXAxis: {
        valueType: 'Category',
        title: 'Years',
        opposedPosition: true,
        isInversed: true
    },
    //Initializing Primary Y Axis
    primaryYAxis: {
        title: 'Exchange rate (INR per USD)',
        edgeLabelPlacement: 'Shift',
        labelIntersectAction: 'Rotate45',
        isInversed: true
    },
    series: [
        {
            type: 'Column',
            dataSource: [
                { x: 2008, y: 15.1 }, { x: 2009, y: 16 }, { x: 2010, y:
21.4 },
                { x: 2011, y: 18 }, { x: 2012, y: 16.2 }, { x: 2013, y:
11 },
                { x: 2014, y: 7.6 }, { x: 2015, y: 1.5 }
            ],
            marker: { dataLabel: { visible: true } },
            xName: 'x',
            yName: 'y', name: 'Years',
        },
    ],
    title: 'Exchange rate',
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Opposed Position

To place an axis opposite from its original position, set [opposedPosition](#) property of the axis to true.

INDEX.JS

```

var chart = new ej.charts.Chart({
  //Initializing Primary X Axis
  primaryXAxis: {
    valueType: 'Category',
    title: 'Years',
    opposedPosition: true,
    isInversed: true
  },
  //Initializing Primary Y Axis
  primaryYAxis: {
    {
      title: 'Exchange rate (INR per USD)',
      edgeLabelPlacement: 'Shift',
      labelIntersectAction: 'Rotate45',
      isInversed: true
    },
    series: [
      {
        type: 'Column',
        dataSource: [
          { x: 2008, y: 15.1 }, { x: 2009, y: 16 }, { x: 2010, y:
21.4 },

```

```

11      { x: 2011, y: 18 }, { x: 2012, y: 16.2 }, { x: 2013, y:
      { x: 2014, y: 7.6 }, { x: 2015, y: 1.5 }
    ],
    marker: { dataLabel: { visible: true }},
    xName: 'x',
    yName: 'y', name: 'Years',
  },
  ],
  title: 'Exchange rate',
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

<!-- markdownlint-disable MD036 -->

Strip line in ##Platform_Name## Chart control

<!-- markdownlint-disable MD036 -->

EJ2 chart supports horizontal and vertical strip lines and customization of stripline in both orientation. To use strip line in axis, we need to inject `StripLine` module using `Chart.Inject(StripLine)` method

Horizontal Strip lines

You can create Horizontal stripline by adding the `stripline` in the vertical axis and set `visible` option to true. Striplines are rendered in the specified start to end range and you can add more than one stripline for an axis.

INDEX.JS

```

var chartData = [
    {x: 1, y: 20}, {x: 2, y: 22}, {x: 3, y: 0}, {x: 4, y: 12}, {x: 5, y: 5},
    {x: 6, y: 15}, {x: 7, y: 6}, {x: 8, y: 12}, {x: 9, y: 20}, {x: 10, y: 7},
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        title: 'Overs'
    },
    primaryYAxis: {
        title: 'Runs',
        stripLines:[
            { start: 15, end: 22, text: 'Good', color: '#ff512f', visible:
true, zIndex: 'Behind', opacity: 0.5 },
            { start: 8, end: 15, text: 'Medium', color: 'pink', opacity:
0.5, visible: true, zIndex: 'Behind' },
            { start: 0, end: 8, text: 'Not enough', color: 'skyblue',
opacity: 0.5, visible: true, zIndex: 'Behind' }]
        },
    series:[{
        dataSource: chartData,
        xName: 'x', yName: 'y',
        type: 'Column',
        marker: { dataLabel: { visible: true}}
    }],
    title: 'India Vs Australia 1st match',
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>

```

```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Vertical Striplines

You can create vertical stripline by adding the `stripline` in the horizontal axis and set `visible` option to `true`. Striplines are rendered in the specified start to end range and you can add more than one stripline for an axis.

INDEX.JS

```
var chartData = [
  {x: 1, y: 20}, {x: 2, y: 22}, {x: 3, y: 0}, {x: 4, y: 12}, {x: 5, y: 5},
  {x: 6, y: 15}, {x: 7, y: 6}, {x: 8, y: 12}, {x: 9, y: 34}, {x: 10, y: 7},
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    title: 'Overs',
    stripLines: [
      {start: 0, end: 5, text: 'powerplay 1', color: 'red', visible:
true, opacity: 0.5, rotation: 45, textStyle: { size: 20, color: 'black' }},
      {start: 5, end: 10, text: 'powerplay 2', color: 'blue', visible:
true, opacity: 0.5, rotation: 45, textStyle: { size: 20, color: 'black' }},
    ]
  },
  primaryYAxis: {
    title: 'Runs'
  },
  series: [{
    dataSource: chartData,
    xName: 'x', yName: 'y',
    type: 'Column',
    marker: { dataLabel: { visible: true } }
  }],
  title: 'India Vs Australia 1st match',
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
```

```

        <div id="element"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customize the strip line

Starting value in specific strip line can be customized by **start** property in strip line. Similarly, ending value is customized by **end**. It can be also set for starting from the corresponding origin of the axis by **startFromOrigin**. Size of the strip line is customized by **size**. Border for the stripline is customized by **border**. Order of the strip line such that whether it should be rendered in behind or over the series elements is customized by **zIndex**.

INDEX.JS

```

var chartData = [
    {x: 1, y: 20}, {x: 2, y: 22}, {x: 3, y: 0}, {x: 4, y: 12}, {x: 5, y: 5},
    {x: 6, y: 15}, {x: 7, y: 6}, {x: 8, y: 12}, {x: 9, y: 34}, {x: 10, y: 7},
];
var chart= new ej.charts.Chart({
    primaryXAxis: {
        title: 'Overs',
        stripLines:[
            { startFromOrigin: true, size: 4, zIndex: 'Behind', opacity:
0.5}
        ]
    },
    primaryYAxis: {
        title: 'Runs'
    },
    series:[{
        dataSource: chartData,
        xName: 'x', yName: 'y',
        type: 'Column',
        marker: { dataLabel:{ visible: true }}
    }],
    title: 'India Vs Australia 1st match',
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

```



```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customize the stripline text

You can customize the text rendered in stripline by `textStyle` property. Rotation of the strip line text can be changed by `rotation` property.

Horizontal and Vertical alignment of stripline text can be changed by `horizontalAlignment` and `verticalAlignment` property.

INDEX.JS

```

var chartData = [
    {x: 1, y: 20}, {x: 2, y: 22}, {x: 3, y: 0}, {x: 4, y: 12}, {x: 5, y: 5},
    {x: 6, y: 15}, {x: 7, y: 6}, {x: 8, y: 12}, {x: 9, y: 34}, {x: 10, y: 7},
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        title: 'Overs',
        stripLines: [
            { startFromOrigin: true, size: 4, zIndex: 'Behind', opacity:
0.5, text: 'Good', verticalAlignment: 'Middle', horizontalAlignment:
'Middle', rotation: 90, textStyle: { size: 15 } },
            { start: 5, end: 8, verticalAlignment: 'Start',
horizontalAlignment: 'End', rotation: 45, text: 'Poor' }
        ]
    },
    primaryYAxis: {
        title: 'Runs'
    },
    series: [{
        dataSource: chartData,
        xName: 'x', yName: 'y',
        type: 'Column',
        marker: { visible: true }
    }],
    title: 'India Vs Australia 1st match',
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Mark the threshold in chart](#)

Multiple panes in ##Platform_Name## Chart control

Chart area can be divided into multiple panes using [rows](#) and [columns](#).

Rows

To split the chart area vertically into number of rows, use [rows](#) property of the chart.

- You can allocate space for each row by using the [height](#) property. The value can be either percentage or in pixel.
- To associate a vertical axis to a particular row, specify its index to [rowIndex](#) property of the axis.
- To customize each row's bottom line, use [border](#) property.

INDEX.JS

```

var chartData = [
  { x: 'Jan', y: 15, y1: 33 }, { x: 'Feb', y: 20, y1: 31 }, { x: 'Mar', y:
  35, y1: 30 },
  { x: 'Apr', y: 40, y1: 28 }, { x: 'May', y: 80, y1: 29 }, { x: 'Jun', y:
  70, y1: 30 },

```

```

    { x: 'Jul', y: 65, y1: 33 }, { x: 'Aug', y: 55, y1: 32 }, { x: 'Sep', y:
50, y1: 34 },
    { x: 'Oct', y: 30, y1: 32 }, { x: 'Nov', y: 35, y1: 32 }, { x: 'Dec', y:
35, y1: 31 }
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        title: 'Months',
        valueType: 'Category',
        interval: 1
    },
    primaryYAxis: {
        minimum: 0, maximum: 90, interval: 20,
        lineStyle: { width: 0 },
        title: 'Temperature (Fahrenheit)',
        labelFormat: '{value}°F'
    },
    // Rows for chart axis
    rows:[
        {
            height: '50%'
        },{
            height: '50%'
        }
    ],
    axes:[
        {
            majorGridLines: { width: 0 },
            rowIndex: 1, opposedPosition: true,
            lineStyle: { width: 0 },
            minimum: 24, maximum: 36, interval: 4,
            name: 'yAxis', title: 'Temperature (Celsius)',
            labelFormat: '{value}°C'
        }
    ],
    series:[{
        dataSource: chartData,
        xName: 'x', yName: 'y',
        name: 'Germany', type: 'Column'
    },{
        dataSource: chartData, width:2,
        xName: 'x', yName: 'y1', yAxisName: 'yAxis',
        name: 'Japan', type: 'Line',
        marker: { visible: true, width: 10, height: 10, border: { width: 2,
color: '#F8AB1D' } }
    }],
    title: 'Weather Condition'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">

```

```

<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

For spanning the vertical axis along multiple row, you can use [span](#) property of an axis.

INDEX.JS

```

var chartData = [
    { x: 'Jan', y: 15, y1: 33 }, { x: 'Feb', y: 20, y1: 31 }, { x: 'Mar', y:
35, y1: 30 },
    { x: 'Apr', y: 40, y1: 28 }, { x: 'May', y: 80, y1: 29 }, { x: 'Jun', y:
70, y1: 30 },
    { x: 'Jul', y: 65, y1: 33 }, { x: 'Aug', y: 55, y1: 32 }, { x: 'Sep', y:
50, y1: 34 },
    { x: 'Oct', y: 30, y1: 32 }, { x: 'Nov', y: 35, y1: 32 }, { x: 'Dec', y:
35, y1: 31 }
];
var chart= new ej.charts.Chart({
    primaryXAxis: {
        title: 'Months',
        valueType: 'Category',
        interval: 1
    },
    primaryYAxis: {
        minimum: 0, maximum: 90, interval: 10,
        lineStyle: { width: 0 },
        title: 'Temperature (Fahrenheit)',
        labelFormat: '{value}°F',
        //Span for chart axis
        span: 2
    },
    rows:[
        {
            height: '50%'
        },{
            height: '50%'
        }
    ]
});

```

```

    }
  ],
  axes:[
    {
      majorGridLines: { width: 0 },
      rowIndex: 1, opposedPosition: true,
      lineStyle: { width: 0 },
      minimum: 24, maximum: 36, interval: 2,
      name: 'yAxis', title: 'Temperature (Celsius)',
      labelFormat: '{value}°C'
    }
  ],
  series:[{
    dataSource: chartData,
    xName: 'x', yName: 'y',
    name: 'Germany', type: 'Column'
  }, {
    dataSource: chartData, width:2,
    xName: 'x', yName: 'y1', yAxisName: 'yAxis',
    name: 'Japan', type: 'Line',
    marker: { visible: true, width: 10, height: 10, border: { width: 2,
color: '#F8AB1D' } }
  }],
  title: 'Weather Condition'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Columns

To split the chart area horizontally into number of columns, use [columns](#) property of the chart.

- You can allocate space for each column by using the [width](#) property. The given width can be either

percentage or in pixel.

- To associate a horizontal axis to a particular column, specify its index to [columnIndex](#) property of the axis.
- To customize each column's bottom line, use [border](#) property.

INDEX.JS

```
var chartData = [
  { x: 'Jan', y: 15, y1: 33 }, { x: 'Feb', y: 20, y1: 31 }, { x: 'Mar', y:
35, y1: 30 },
  { x: 'Apr', y: 40, y1: 28 }, { x: 'May', y: 80, y1: 29 }, { x: 'Jun', y:
70, y1: 30 },
  { x: 'Jul', y: 65, y1: 33 }, { x: 'Aug', y: 55, y1: 32 }, { x: 'Sep', y:
50, y1: 34 },
  { x: 'Oct', y: 30, y1: 32 }, { x: 'Nov', y: 35, y1: 32 }, { x: 'Dec', y:
35, y1: 31 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    valueType: 'Category',
    interval: 1
  },
  primaryYAxis: {
    minimum: 0, maximum: 90, interval: 10,
    lineStyle: { width: 0 },
    title: 'Temperature (Fahrenheit)',
    labelFormat: '{value}°F'
  },
  // Columns for chart axis
  columns: [
    {
      width: '50%'
    }, {
      width: '50%'
    }
  ],
  axes: [
    {
      majorGridLines: { width: 0 },
      columnIndex: 1,
      valueType: 'Category',
      lineStyle: { width: 0 },
      name: 'xAxis'
    }
  ],
  series: [{
    dataSource: chartData,
```

```

        xName: 'x', yName: 'y',
        name: 'Germany', type: 'Column'
    }, {
        dataSource: chartData, width: 2,
        xName: 'x', yName: 'y1', xAxisName: 'xAxis',
        name: 'Japan', type: 'Line',
        marker: { visible: true, width: 10, height: 10, border: { width: 2,
color: '#F8AB1D' } }
    }],
    title: 'Weather Condition'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

For spanning the vertical axis along multiple column, you can use [span](#) property of an axis.

INDEX.JS

```

var chartData = [
  { x: 'Jan', y: 15, y1: 33 }, { x: 'Feb', y: 20, y1: 31 }, { x: 'Mar', y:
35, y1: 30 },
  { x: 'Apr', y: 40, y1: 28 }, { x: 'May', y: 80, y1: 29 }, { x: 'Jun', y:
70, y1: 30 },
  { x: 'Jul', y: 65, y1: 33 }, { x: 'Aug', y: 55, y1: 32 }, { x: 'Sep', y:
50, y1: 34 },
  { x: 'Oct', y: 30, y1: 32 }, { x: 'Nov', y: 35, y1: 32 }, { x: 'Dec', y:
35, y1: 31 }
];

```

```

var chart = new ej.charts.Chart({
  primaryXAxis: {
    valueType: 'Category',
    interval: 1,
    // Span for chart axis
    span: 2
  },
  primaryYAxis: {
    minimum: 0, maximum: 90, interval: 10,
    lineStyle: { width: 0 },
    title: 'Temperature (Fahrenheit)',
    labelFormat: '{value}°F'
  },
  columns:[
    {
      width: '50%'
    },{
      width: '50%'
    }
  ],
  axes:[
    {
      valueType: 'Category',
      majorGridLines: { width: 0 },
      columnIndex: 1, opposedPosition: true,
      lineStyle: { width: 0 },
      name: 'xAxis'
    }
  ],
  series:[{
    dataSource: chartData,
    xName: 'x', yName: 'y',
    name: 'Germany', type: 'Column'
  },{
    dataSource: chartData, width:2,
    xName: 'x', yName: 'y1', xAxisName: 'xAxis',
    name: 'Japan', type: 'Line',
    marker: { visible: true, width: 10, height: 10, border: { width: 2,
color: '#F8AB1D' } }
  }],
  title: 'Weather Condition'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>

```



```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Chart Types

Line Chart in ##Platform_Name## control

Line

To render a line series, use series [type](#) as `Line` and inject `LineSeries` module using `Chart.Inject(LineSeries)` method.

INDEX.JS

```

var chartData = [
    { x: 2005, y: 28 }, { x: 2006, y: 25 }, { x: 2007, y: 26 }, { x: 2008, y:
27 },
    { x: 2009, y: 32 }, { x: 2010, y: 35 }, { x: 2011, y: 30 }
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        title: 'Year',
        minimum: 2004, maximum: 2012, interval: 1
    },
    primaryYAxis: {
        minimum: 20, maximum: 40, interval: 5,
        title: 'Efficiency',
        labelFormat: '{value}%'
    },
    series:[{
        dataSource: chartData, width:2,
        xName: 'x', yName: 'y',
        name: 'India',
        //Series type as line
        type: 'Line'
    }],
    title: 'Efficiency of oil-fired power production'
}, '#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>

```

```

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Multicolored line

To render a multicolored line series, use the series type as `MultiColoredLine`, and inject the `MultiColoredLineSeries` module using `Chart.Inject(MultiColoredLineSeries)` method. Here, the individual colors to the data can be mapped by using `pointColorMapping`.

INDEX.JS

```

var chartData = [
    { x: 2005, y: 28 }, { x: 2006, y: 25 }, { x: 2007, y: 26 }, { x: 2008, y:
27 },
    { x: 2009, y: 32 }, { x: 2010, y: 35 }, { x: 2011, y: 30 }
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        title: 'Year',
        minimum: 2004, maximum: 2012, interval: 1
    },
    primaryYAxis: {
        minimum: 20, maximum: 40, interval: 5,
        title: 'Efficiency',
        labelFormat: '{value}%'
    },
    series:[{
        dataSource: chartData, width:2,
        xName: 'x', yName: 'y',
        name: 'India',
        //Series type as line
        type: 'Line'
    }],
    title: 'Efficiency of oil-fired power production'

```

```
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Series customization

The following properties can be used to customize the **line** series.

- [fill](#) – Specifies the color of the series.
- [opacity](#) – Specifies the opacity of [fill](#).
- [dashArray](#) – Specifies the dashes for series.
- [width](#) – Specifies the width for series.

INDEX.JS

```
var chartData = [
  { x: 2005, y: 28 }, { x: 2006, y: 25 }, { x: 2007, y: 26 }, { x: 2008, y:
27 },
  { x: 2009, y: 32 }, { x: 2010, y: 35 }, { x: 2011, y: 30 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    title: 'Year',
    minimum: 2004, maximum: 2012, interval: 1
  },
  primaryYAxis: {
    minimum: 20, maximum: 40, interval: 5,
```

```

        title: 'Efficiency',
        labelFormat: '{value}%'
    },
    series:[{
        dataSource: chartData,
        //fill for chart series
        fill: 'red',
        //line width as 4 for chart series
        width:4,
        //dash array value as 5,5
        dashArray: '5,5',
        xName: 'x', yName: 'y',
        name: 'India', type: 'Line'
    }],
    title: 'Efficiency of oil-fired power production'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Data label](#)
- [Tooltip](#)

Step line Chart in ##Platform_Name## control

Step line

To render a step line series, use series `type` as `StepLine` and inject `StepLineSeries` module using `Chart.Inject(StepLineSeries)` method.

INDEX.JS

```
var chartData = [
  { x: 2006, y: 378 }, { x: 2007, y: 416 },
  { x: 2008, y: 404 }, { x: 2009, y: 390 },
  { x: 2010, y: 376 }, { x: 2011, y: 365 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    title: 'Year',
    minimum: 2005, maximum: 2012, interval: 1
  },
  primaryYAxis: {
    minimum: 330, maximum: 450, interval: 30,
    title: 'Intensity (g/kWh)'
  },
  series:[{
    dataSource: chartData, width:2,
    xName: 'x', yName: 'y',
    name: 'USA',
    // Series type as StepLine
    type: 'StepLine'
  }],
  title: 'CO2 - Intensity Analysis'
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
```

```

}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Series customization

The following properties can be used to customize the **step line** series.

- [fill](#) – Specifies the color of the series.
- [opacity](#) – Specifies the opacity of [fill](#).
- [dashArray](#) – Specifies the dashes for series.
- [width](#) – Specifies the width for series.
- [step](#) – Specifies the position of the step for the series.

INDEX.JS

```

var chartData = [
  { x: 2005, y: 28 }, { x: 2006, y: 25 }, { x: 2007, y: 26 }, { x: 2008, y:
27 },
  { x: 2009, y: 32 }, { x: 2010, y: 35 }, { x: 2011, y: 30 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    title: 'Year',
    minimum: 2004, maximum: 2012, interval: 1
  },
  primaryYAxis: {
    minimum: 20, maximum: 40, interval: 5,
    title: 'Efficiency',
    labelFormat: '{value}%'
  },
  series:[{
    dataSource: chartData,
    //fill for chart series
    fill: 'red',
    //line width as 4 for chart series
    width:4,
    //dash array value as 5,5
    dashArray: '5,5',
    xName: 'x', yName: 'y',
    type: 'StepLine',
    opacity: 0.5,
    step: 'Left'
  }],
  title: 'Efficiency of oil-fired power production'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">

```

```

<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See also

- [Data label](#)
- [Tooltip](#)

Stack line Chart in ##Platform_Name## control

Stacked Line

To render a stacked line series, use series [type](#) as `StackingLine` and inject `StackingLineSeries` module using `Chart.Inject(StackingLineSeries)` method.

INDEX.JS

```

var chartData = [
    { x: 'Food', y: 90, y1: 40, y2: 70, y3: 120 },
    { x: 'Transport', y: 80, y1: 90, y2: 110, y3: 70 },
    { x: 'Medical', y: 50, y1: 80, y2: 120, y3: 50 },
    { x: 'Clothes', y: 70, y1: 30, y2: 60, y3: 180 },
    { x: 'Personal Care', y: 30, y1: 80, y2: 80, y3: 30 },
    { x: 'Books', y: 10, y1: 40, y2: 30, y3: 270 },
    { x: 'Fitness', y: 100, y1: 30, y2: 70, y3: 40 },
    { x: 'Electricity', y: 55, y1: 95, y2: 55, y3: 75 },
    { x: 'Tax', y: 20, y1: 50, y2: 40, y3: 65 },
    { x: 'Pet Care', y: 40, y1: 20, y2: 80, y3: 95 },
    { x: 'Education', y: 45, y1: 15, y2: 45, y3: 195 },
    { x: 'Entertainment', y: 75, y1: 45, y2: 65, y3: 115 }
];

var chart = new ej.charts.Chart({
    //Initializing Primary X Axis
    primaryXAxis: {
        interval: 1,

```

```

        valueType: 'Category'
    },
    //Initializing Primary Y Axis
    primaryYAxis:
    {
        interval: 100,
    },
    chartArea: { border: { width: 0 } },
    //Initializing Chart Series
    series: [
        {
            type: 'StackingLine', dataSource: chartData, marker: {
visible: true },
            dashArray: '5, 1', xName: 'x', width: 2, yName: 'y', name:
            'John'
        },
        {
            type: 'StackingLine', dataSource: chartData, marker: {
visible: true },
            dashArray: '5, 1', xName: 'x', width: 2, yName: 'y1', name:
            'Peter'
        },
        {
            type: 'StackingLine', dataSource: chartData, marker: {
visible: true },
            dashArray: '5, 1', xName: 'x', width: 2, yName: 'y2', name:
            'Steve'
        },
        {
            type: 'StackingLine', dataSource: chartData, marker: {
visible: true },
            dashArray: '5, 1', xName: 'x', width: 2, yName: 'y3', name:
            'Charle'
        }
    ],
    //Initializing User Interaction Tooltip
    tooltip: {
        enable: true
    },
});
chart.appendTo('#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
```



```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Series customization

The following properties can be used to customize the **stacked line** series.

- [fill](#) – Specifies the color of the series.
- [opacity](#) – Specifies the opacity of [fill](#).
- [dashArray](#) – Specifies the dashes for series.
- [width](#) – Specifies the width for series.

INDEX.JS

```

var chartData = [
    { x: 2005, y: 28 }, { x: 2006, y: 25 }, { x: 2007, y: 26 }, { x: 2008, y:
27 },
    { x: 2009, y: 32 }, { x: 2010, y: 35 }, { x: 2011, y: 30 }
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        title: 'Year',
        minimum: 2004, maximum: 2012, interval: 1
    },
    primaryYAxis: {
        minimum: 20, maximum: 40, interval: 5,
        title: 'Efficiency',
        labelFormat: '{value}%'
    },
    series:[{
        dataSource: chartData,
        //fill for chart series
        fill: 'red',
        //line width as 4 for chart series
        width:4,
        //dash array value as 5,5
        dashArray: '5,5',
        xName: 'x', yName: 'y',
        marker:{ visible: true, isFilled: true, height: 10, width:10},
        name: 'India', type: 'StackingLine'
    }],
});

```

```

    title: 'Efficiency of oil-fired power production'
  }, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Data label](#)
- [Tooltip](#)

Stacked line Chart in ##Platform_Name## control

100% Stacked Line

To render a 100% stacked line series, use series [type](#) as `StackingLine100` and inject `StackingLineSeries` module using `Chart.Inject(StackingLineSeries)` method.

INDEX.JS

```

var chartData = [
  { x: 'Food', y: 90, y1: 40, y2: 70, y3: 120 },
  { x: 'Transport', y: 80, y1: 90, y2: 110, y3: 70 },
  { x: 'Medical', y: 50, y1: 80, y2: 120, y3: 50 },
  { x: 'Clothes', y: 70, y1: 30, y2: 60, y3: 180 },
  { x: 'Personal Care', y: 30, y1: 80, y2: 80, y3: 30 },
  { x: 'Books', y: 10, y1: 40, y2: 30, y3: 270 },
  { x: 'Fitness', y: 100, y1: 30, y2: 70, y3: 40 },
  { x: 'Electricity', y: 55, y1: 95, y2: 55, y3: 75 },

```

```

    { x: 'Tax', y: 20, y1: 50, y2: 40, y3: 65 },
    { x: 'Pet Care', y: 40, y1: 20, y2: 80, y3: 95 },
    { x: 'Education', y: 45, y1: 15, y2: 45, y3: 195 },
    { x: 'Entertainment', y: 75, y1: 45, y2: 65, y3: 115 }
  ];

  var chart = new ej.charts.Chart({
    //Initializing Primary X Axis
    primaryXAxis: {
      interval: 1, valueType: 'Category'
    },
    primaryYAxis: {
      interval: 20
    },
    chartArea: { border: { width: 0 } },
    series: [
      {
        type: 'StackingLine100', dataSource: chartData, marker: {
visible: true },
        dashArray: '5, 1', xName: 'x', width: 2, yName: 'y', name:
        'John'
      },
      {
        type: 'StackingLine100', dataSource: chartData, marker: {
visible: true },
        dashArray: '5, 1', xName: 'x', width: 2, yName: 'y1', name:
        'Peter'
      },
      {
        type: 'StackingLine100', dataSource: chartData, marker: {
visible: true },
        dashArray: '5, 1', xName: 'x', width: 2, yName: 'y2', name:
        'Steve'
      },
      {
        type: 'StackingLine100', dataSource: chartData, marker: {
visible: true },
        dashArray: '5, 1', xName: 'x', width: 2, yName: 'y3', name:
        'Charle'
      }
    ],
    tooltip: {
      enable: true,
      format: '${point.x} : <b>${point.y} (${point.percentage}%</b>'
    },
  });
  chart.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">

```

```

<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Series customization

The following properties can be used to customize the **100% stacked line** series.

- [fill](#) – Specifies the color of the series.
- [opacity](#) – Specifies the opacity of [fill](#).
- [dashArray](#) – Specifies the dashes for series.
- [width](#) – Specifies the width for series.

INDEX.JS

```

var data = [
    { x: 2005, y: 90, y1: 40, y2: 70, y3: 120 },
    { x: 2006, y: 80, y1: 90, y2: 110, y3: 70 },
    { x: 2007, y: 50, y1: 80, y2: 120, y3: 50 },
    { x: 2008, y: 70, y1: 30, y2: 60, y3: 180 },
    { x: 2009, y: 30, y1: 80, y2: 80, y3: 30 },
    { x: 2010, y: 10, y1: 40, y2: 30, y3: 270 },
    { x: 2011, y: 100, y1: 30, y2: 70, y3: 40 }
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        title: 'Year',
        minimum: 2004, maximum: 2012, interval: 1
    },
    primaryYAxis: {
        title: 'Efficiency',
        labelFormat: '{value}%',
    },
    series:[{
        dataSource: data,
        fill: 'green',width: 2,

```

```

        dashArray: '2', xName: 'x', marker: {visible: true},
        yName: 'y', type: 'StackingLine100',
    },
    {
        dataSource: data,
        fill: 'pink', width: 2,
        dashArray: '2', xName: 'x', marker: {visible: true},
        yName: 'y1', type: 'StackingLine100',
    },
    {
        dataSource: data,
        fill: 'yellow', width: 2,
        dashArray: '2', xName: 'x', marker: {visible: true},
        yName: 'y2', type: 'StackingLine100',
    },
    {
        dataSource: data,
        fill: 'red', width: 2,
        dashArray: '2', xName: 'x', marker: {visible: true},
        yName: 'y3', type: 'StackingLine100',
    }
  ],
  title: 'Efficiency of oil-fired power production'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

[See Also](#)

- [Data label](#)
- [Tooltip](#)

Spline Chart in ##Platform_Name## control

[Spline](#)

To render a spline series, use series [type](#) as `Spline` and inject `SplineSeries` module using `Chart.Inject(SplineSeries)` method.

INDEX.JS

```
var chartData = [
  { x: 'Jan', y: -1 }, { x: 'Feb', y: -1 }, { x: 'Mar', y: 2 },
  { x: 'Apr', y: 8 }, { x: 'May', y: 13 }, { x: 'Jun', y: 18 },
  { x: 'Jul', y: 21 }, { x: 'Aug', y: 20 }, { x: 'Sep', y: 16 },
  { x: 'Oct', y: 10 }, { x: 'Nov', y: 4 }, { x: 'Dec', y: 0 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    title: 'Month',
    valueType: 'Category'
  },
  primaryYAxis: {
    minimum: -5, maximum: 35, interval: 5,
    title: 'Temperature in Celsius',
    labelFormat: '{value}C'
  },
  series:[{
    dataSource: chartData, width:2,
    xName: 'x', yName: 'y',
    name: 'London',
    // Series type as spline series
    type: 'Spline'
  }],
  title: 'Climate Graph-2012'
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>
```

```

    <div id="container">
      <div id="element"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Series customization

The following properties can be used to customize the **spline** series.

- [fill](#) – Specifies the color of the series.
- [opacity](#) – Specifies the opacity of [fill](#).
- [dashArray](#) – Specifies the dashes for series.
- [width](#) – Specifies the width for series.

INDEX.JS

```

var chartData = [
  { x: 2005, y: 28 }, { x: 2006, y: 25 }, { x: 2007, y: 26 }, { x: 2008, y:
27 },
  { x: 2009, y: 32 }, { x: 2010, y: 35 }, { x: 2011, y: 30 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    title: 'Year',
    minimum: 2004, maximum: 2012, interval: 1
  },
  primaryYAxis: {
    minimum: 20, maximum: 40, interval: 5,
    title: 'Efficiency',
    labelFormat: '{value}%'
  },
  series:[{
    dataSource: chartData,
    //fill for chart series
    fill: 'red',
    //line width as 4 for chart series
    width:4,
    //dash array value as 5,5
    dashArray: '5,5',
    xName: 'x', yName: 'y',
    type: 'Spline'
  }],
  title: 'Efficiency of oil-fired power production'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Data label](#)
- [Tooltip](#)

Area Chart in ##Platform_Name## control

Area

INDEX.JS

```

var chartData = [
  { x: 1900, y: 4 }, { x: 1920, y: 3.0 }, { x: 1940, y: 3.8 },
  { x: 1960, y: 3.4 }, { x: 1980, y: 3.2 }, { x: 2000, y: 3.9 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    title: 'Year',
    minimum: 1900, maximum: 2000, interval: 10,
    edgeLabelPlacement: 'Shift'
  },
  primaryYAxis: {
    minimum: 2, maximum: 5, interval: 0.5,
    title: 'Sales Amount in Millions'
  },
  series:[{
    dataSource: chartData,
    xName: 'x', yName: 'y',

```



```

        opacity: 0.5, fill: '#69D2E7',
        name: 'Product A', border: { width: 2 },
        // Series type as area series
        type: 'Area'
    }],
    title: 'Average Sales Comparison'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Multicolored area

To render a multicolored area series, use the series type as `MultiColoredArea`, and inject the `MultiColoredAreaSeries` module using `Chart.Inject(MultiColoredAreaSeries)` method. The required segments of the series can be customized using the `value`, `color`, and `dashArray`.

INDEX.JS

```

var chart = new ej.charts.Chart({
  series:[{
    dataSource: [{ x: 2005, y: 26 }, { x: 2006, y: 25}, { x: 2007, y:
26 }, { x: 2008, y: 27 },
    { x: 2009, y: 32}, { x: 2010, y: 35 }, { x: 2011, y: 25 }],
    xName: 'x', yName: 'y',
    type: 'MultiColoredArea',
    segmentAxis: 'X',
    segments: [{
      value: 2007,

```

```

        color: 'blue'
    }, {
        value: 2009,
        color: 'lightgreen'
    }, {
        color: 'orange'
    }
  ],
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Series customization

The following properties can be used to customize the **area** series.

- [fill](#) – Specifies the color of the area series.
- [opacity](#) – Specifies the opacity of [fill](#).
- [dashArray](#) – Specifies the dashes of series.

INDEX.JS

```

var chartData = [
  { x: 1900, y: 4 }, { x: 1920, y: 3.0 }, { x: 1940, y: 3.8 },
  { x: 1960, y: 3.4 }, { x: 1980, y: 3.2 }, { x: 2000, y: 3.9 }
];
var chart = new ej.charts.Chart({

```

```

primaryXAxis: {
    title: 'Year',
    minimum: 1900, maximum: 2000, interval: 10,
    edgeLabelPlacement: 'Shift'
},
primaryYAxis: {
    minimum: 2, maximum: 5, interval: 0.5,
    title: 'Sales Amount in Millions'
},
series:[{
    dataSource: chartData,
    xName: 'x', yName: 'y',
    opacity: 0.5, fill:'#69D2E7',
    //stroke-width for chart series
    width: 3,
    // dashArray for chart series
    dashArray: '5,5',
    name: 'Product A',
    // Series type as area series
    type: 'Area',
    border:{width:2, color:'Red'},
}],
title: 'Average Sales Comparison'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Area border

The following properties can be used to customize the **area** series.

- [fill](#) – Specifies the color of the area series.
- [opacity](#) – Specifies the opacity of [fill](#).
- [dashArray](#) – Specifies the dashes of series.

INDEX.JS

```

var chartData = [
  { x: 1900, y: 4 }, { x: 1920, y: 3.0 }, { x: 1940, y: 3.8 },
  { x: 1960, y: 3.4 }, { x: 1980, y: 3.2 }, { x: 2000, y: 3.9 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    title: 'Year',
    minimum: 1900, maximum: 2000, interval: 10,
    edgeLabelPlacement: 'Shift'
  },
  primaryYAxis: {
    minimum: 2, maximum: 5, interval: 0.5,
    title: 'Sales Amount in Millions'
  },
  series:[{
    dataSource: chartData,
    xName: 'x', yName: 'y',
    opacity: 0.5, fill: '#69D2E7',
    name: 'Product A', border: { width: 2 },
    // Series type as area series
    type: 'Area'
  }],
  title: 'Average Sales Comparison'
}, '#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
```

```

<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Data label](#)
- [Tooltip](#)

Range area Chart in ##Platform_Name## control

Range area

To render a range area series, use series [type](#) as `RangeArea` and inject `RangeAreaSeries` module using `Chart.Inject(RangeAreaSeries)` method.

Since the `RangeArea` series requires two y values for a point, you have to add the high and low value. High and Low value specifies the maximum and minimum range of the points.

INDEX.JS

```

var series= [];
var value = 70;
var point;
for (var i = 1; i < 70; i++) {
    if (Math.random() > .5) {
        value += Math.random();
    } else {
        value -= Math.random();
    }
    point = { x: new Date(1930 + i, 5, i), high: value, low: value - 14 };
    series.push(point);
}
var chart = new ej.charts.Chart({
    primaryXAxis: {
        title: 'Month',
        valueType: 'DateTime',
        edgeLabelPlacement: 'Shift',
    },
    primaryYAxis: {
        title: 'Temperature(Celsius)',
        minimum: 50, maximum: 80, interval: 5,
    },
    series: [
        {
            type: 'RangeArea',
            name: 'India',
            dataSource: series,
            xName: 'x', high: 'high', low: 'low', opacity: 0.5,
            fill: '#69D2E7', border: { color: 'blueviolet', width: 1 }
        }
    ]
});

```

```
    }],
    title: 'Maximum and Minimum Temperature',
  }, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Series customization

The following properties can be used to customize the **range area** series.

- [fill](#) – Specifies the color of the area series.
- [opacity](#) – Specifies the opacity of [fill](#).
- [dashArray](#) – Specifies the dashes of series.

INDEX.JS

```
var series= [];
var value = 70;
var point;
for (var i = 1; i < 70; i++) {
  if (Math.random() > .5) {
    value += Math.random();
  } else {
    value -= Math.random();
  }
  point = { x: new Date(1930 + i, 5, i), high: value, low: value - 14 };
  series.push(point);
}
```

```

}
var chart = new ej.charts.Chart({
  primaryXAxis: {
    title: 'Month',
    valueType: 'DateTime',
    edgeLabelPlacement: 'Shift',
  },
  primaryYAxis:
  {
    title: 'Temperature(Celsius)',
    minimum: 50, maximum: 80, interval: 5,
  },
  series: [
    {
      type: 'RangeArea',
      name: 'India',
      dataSource: series,
      xName: 'x', high: 'high', low: 'low', opacity: 0.5,
      opacity: 0.7,
      dashArray: '4', fill: 'grey', border: { color: 'blueviolet', width:
1 }
    }
  ],
  title: 'Maximum and Minimum Temperature',
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Data label](#)
- [Tooltip](#)

Range step area Chart in ##Platform_Name## control

Range step area

To render the range step area series, use the series [type](#) as a `RangeStepArea` and inject the `RangeStepAreaSeries` module using the `Chart.Inject(RangeStepAreaSeries)` method.

INDEX.JS

```
var splinedata = [
  { x: 'Jan', high: 14, low: 4, high1: 29, low1: 19 },
  { x: 'Feb', high: 17, low: 7, high1: 32, low1: 22 },
  { x: 'Mar', high: 20, low: 10, high1: 35, low1: 25 },
  { x: 'Apr', high: 22, low: 12, high1: 37, low1: 27 },
  { x: 'May', high: 20, low: 10, high1: 35, low1: 25 },
  { x: 'Jun', high: 17, low: 7, high1: 32, low1: 22 },
  { x: 'Jul', high: 15, low: 5, high1: 30, low1: 20 },
  { x: 'Aug', high: 17, low: 7, high1: 32, low1: 22 },
  { x: 'Sep', high: 20, low: 10, high1: 35, low1: 25 },
  { x: 'Oct', high: 22, low: 12, high1: 37, low1: 27 },
  { x: 'Nov', high: 20, low: 10, high1: 35, low1: 25 },
  { x: 'Dec', high: 17, low: 7, high1: 32, low1: 22 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    valueType: 'Category',
    edgeLabelPlacement: 'Shift',
    majorGridLines: { width: 0 },
  },
  primaryYAxis: {
    labelFormat: '{value}°C',
    lineStyle: { width: 0 },
    minimum: 0,
    maximum: 40,
    majorTickLines: { width: 0 }
  },
  series: [
    {
      type: 'RangeStepArea',
      name: 'India',
      dataSource: splinedata,
      xName: 'x', high: 'high1', low: 'low1',
      opacity: 0.4,
    },
  ],
  title: 'Monthly Temperature Range',
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
```



```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Series customization

The following properties can be used to customize the **area** series.

- [fill](#) – Specifies the color of the area series.
- [opacity](#) – Specifies the opacity of [fill](#).
- [dashArray](#) – Specifies the dashes of series.
- [step](#) – Specifies the position of the step for the series.

INDEX.JS

```

var splinedata = [
    { x: 'Jan', high: 14, low: 4, high1: 29, low1: 19 },
    { x: 'Feb', high: 17, low: 7, high1: 32, low1: 22 },
    { x: 'Mar', high: 20, low: 10, high1: 35, low1: 25 },
    { x: 'Apr', high: 22, low: 12, high1: 37, low1: 27 },
    { x: 'May', high: 20, low: 10, high1: 35, low1: 25 },
    { x: 'Jun', high: 17, low: 7, high1: 32, low1: 22 },
    { x: 'Jul', high: 15, low: 5, high1: 30, low1: 20 },
    { x: 'Aug', high: 17, low: 7, high1: 32, low1: 22 },
    { x: 'Sep', high: 20, low: 10, high1: 35, low1: 25 },
    { x: 'Oct', high: 22, low: 12, high1: 37, low1: 27 },
    { x: 'Nov', high: 20, low: 10, high1: 35, low1: 25 },
    { x: 'Dec', high: 17, low: 7, high1: 32, low1: 22 }
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        valueType: 'Category',
        edgeLabelPlacement: 'Shift',
        majorGridLines: { width: 0 },

```

```

    },
    primaryYAxis: {
        labelFormat: '{value}°C',
        lineStyle: { width: 0 },
        minimum: 0,
        maximum: 40,
        majorTickLines: { width: 0 }
    },
    series: [
        {
            type: 'RangeStepArea',
            name: 'India',
            dataSource: splinedata,
            xName: 'x', high: 'high1', low: 'low1', dashArray: '5,5',
            opacity: 0.4, fill: 'red', border: { width: 2, color: 'blue'},
            step: 'Center'
        }
    ],
    title: 'Monthly Temperature Range',
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

See also

- [Data label](#)
- [Tooltip](#)

Spline range area Chart in `##Platform_Name##` control

Spline Range Area

The Spline Range Area Chart is used to display continuous data points as a set of splines that vary between high and low values over intervals of time and across different categories.

To render a spline range area series, use series [type](#) as `SplineRangeArea` and inject `SplineRangeAreaSeries` module using `Chart.Inject(SplineRangeAreaSeries)` method.

INDEX.JS

```
var splinedata = [
  { x: 'Jan', high: 14, low: 4, high1: 29, low1: 19 },
  { x: 'Feb', high: 17, low: 7, high1: 32, low1: 22 },
  { x: 'Mar', high: 20, low: 10, high1: 35, low1: 25 },
  { x: 'Apr', high: 22, low: 12, high1: 37, low1: 27 },
  { x: 'May', high: 20, low: 10, high1: 35, low1: 25 },
  { x: 'Jun', high: 17, low: 7, high1: 32, low1: 22 },
  { x: 'Jul', high: 15, low: 5, high1: 30, low1: 20 },
  { x: 'Aug', high: 17, low: 7, high1: 32, low1: 22 },
  { x: 'Sep', high: 20, low: 10, high1: 35, low1: 25 },
  { x: 'Oct', high: 22, low: 12, high1: 37, low1: 27 },
  { x: 'Nov', high: 20, low: 10, high1: 35, low1: 25 },
  { x: 'Dec', high: 17, low: 7, high1: 32, low1: 22 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    valueType: 'Category',
    edgeLabelPlacement: 'Shift',
    majorGridLines: { width: 0 },
  },
  primaryYAxis: {
    labelFormat: '{value}°C',
    lineStyle: { width: 0 },
    minimum: 0,
    maximum: 40,
    majorTickLines: { width: 0 }
  },
  series: [
    {
      type: 'SplineRangeArea',
      name: 'England',
      dataSource: splinedata,
      xName: 'x', high: 'high', low: 'low',
      opacity: 0.4,
    },
    {
      type: 'SplineRangeArea',
      name: 'India',
      dataSource: splinedata,
      xName: 'x', high: 'high1', low: 'low1',
      opacity: 0.4,
    }
  ],
  title: 'Monthly Temperature Range',
}, '#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Series customization

The following properties can be used to customize the **spline range area** series.

- [fill](#) – Specifies the color of the area series.
- [opacity](#) – Specifies the opacity of [fill](#).
- [dashArray](#) – Specifies the dashes of series.

INDEX.JS

```

var splinedata = [
  { x: 'Jan', high: 14, low: 4, high1: 29, low1: 19 },
  { x: 'Feb', high: 17, low: 7, high1: 32, low1: 22 },
  { x: 'Mar', high: 20, low: 10, high1: 35, low1: 25 },
  { x: 'Apr', high: 22, low: 12, high1: 37, low1: 27 },
  { x: 'May', high: 20, low: 10, high1: 35, low1: 25 },
  { x: 'Jun', high: 17, low: 7, high1: 32, low1: 22 },
  { x: 'Jul', high: 15, low: 5, high1: 30, low1: 20 },
  { x: 'Aug', high: 17, low: 7, high1: 32, low1: 22 },
  { x: 'Sep', high: 20, low: 10, high1: 35, low1: 25 },
  { x: 'Oct', high: 22, low: 12, high1: 37, low1: 27 },
  { x: 'Nov', high: 20, low: 10, high1: 35, low1: 25 },
  { x: 'Dec', high: 17, low: 7, high1: 32, low1: 22 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {

```

```

        valueType: 'Category',
        edgeLabelPlacement: 'Shift',
        majorGridLines: { width: 0 },
    },
    primaryYAxis: {
        labelFormat: '{value}°C',
        lineStyle: { width: 0 },
        minimum: 0,
        maximum: 40,
        majorTickLines: { width: 0 }
    },
    series: [
        {
            type: 'SplineRangeArea',
            name: 'England',
            dataSource: splinedata,
            xName: 'x', high: 'high', low: 'low',
            opacity: 0.4, border:{ width: 2, color: 'red'},
            dashArray: '3'
        },
        {
            type: 'SplineRangeArea',
            name: 'India',
            dataSource: splinedata,
            xName: 'x', high: 'high1', low: 'low1',
            opacity: 0.4, border:{ width: 2, color: 'red'},
            dashArray: '3'
        }
    ],
    title: 'Monthly Temperature Range',
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";

```

```

}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Data label](#)
- [Tooltip](#)

Stack area Chart in ##Platform_Name## control

Stacked Area

To render a stacked area series, use series [type](#) as `StackingArea` and inject `StackingAreaSeries` module using `Chart.Inject(StackingAreaSeries)` method.

INDEX.JS

```

var chartData = [
  { x: new Date(2000, 0, 1), y: 0.61, y1: 0.03, y2: 0.48, y3: 0.23 },
  { x: new Date(2001, 0, 1), y: 0.81, y1: 0.05, y2: 0.53, y3: 0.17 },
  { x: new Date(2002, 0, 1), y: 0.91, y1: 0.06, y2: 0.57, y3: 0.17 },
  { x: new Date(2003, 0, 1), y: 1, y1: 0.09, y2: 0.61, y3: 0.20 },
  { x: new Date(2004, 0, 1), y: 1.19, y1: 0.14, y2: 0.63, y3: 0.23 },
  { x: new Date(2005, 0, 1), y: 1.47, y1: 0.20, y2: 0.64, y3: 0.36 },
  { x: new Date(2006, 0, 1), y: 1.74, y1: 0.29, y2: 0.66, y3: 0.43 },
  { x: new Date(2007, 0, 1), y: 1.98, y1: 0.46, y2: 0.76, y3: 0.52 },
  { x: new Date(2008, 0, 1), y: 1.99, y1: 0.64, y2: 0.77, y3: 0.72 },
  { x: new Date(2009, 0, 1), y: 1.70, y1: 0.75, y2: 0.55, y3: 1.29 },
  { x: new Date(2010, 0, 1), y: 1.48, y1: 1.06, y2: 0.54, y3: 1.38 },
  { x: new Date(2011, 0, 1), y: 1.38, y1: 1.25, y2: 0.57, y3: 1.82 },
  { x: new Date(2012, 0, 1), y: 1.66, y1: 1.55, y2: 0.61, y3: 2.16 },
  { x: new Date(2013, 0, 1), y: 1.66, y1: 1.55, y2: 0.67, y3: 2.51 },
  { x: new Date(2014, 0, 1), y: 1.67, y1: 1.65, y2: 0.67, y3: 2.61 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    title: 'Years',
    valueType: 'DateTime',
    intervalType: 'Years',
    majorTickLines: { width: 0 },
    labelFormat: 'y',
    edgeLabelPlacement: 'Shift'
  },
  primaryYAxis: {
    title: 'Spend in Billions',
    minimum: 0,
    maximum: 7,
    interval: 1,
    majorTickLines: { width: 0 },
    labelFormat: '{value}B'
  },
  series: [
    {

```

```

        dataSource: chartData, xName: 'x', yName: 'y',
        //Series type as stacked area series
        type: 'StackingArea',
        name: 'Organic',
    }, {
        dataSource: chartData, xName: 'x', yName: 'y1',
        type: 'StackingArea', name: 'Fair-trade',
    }, {
        dataSource: chartData, xName: 'x', yName: 'y2',
        type: 'StackingArea', name: 'Veg Alternatives',
    }, {
        dataSource: chartData, xName: 'x', yName: 'y3',
        type: 'StackingArea', name: 'Others',
    }
    ],
    title: 'Trend in Sales of Ethical Produce'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Series customization

The following properties can be used to customize the **stacked area** series.

- [fill](#) – Specifies the color of the area series.
- [opacity](#) – Specifies the opacity of [fill](#).
- [dashArray](#) – Specifies the dashes of series.

INDEX.JS

```

var chartData = [
  { x: new Date(2000, 0, 1), y: 0.61, y1: 0.03, y2: 0.48, y3: 0.23 },
  { x: new Date(2001, 0, 1), y: 0.81, y1: 0.05, y2: 0.53, y3: 0.17 },
  { x: new Date(2002, 0, 1), y: 0.91, y1: 0.06, y2: 0.57, y3: 0.17 },
  { x: new Date(2003, 0, 1), y: 1, y1: 0.09, y2: 0.61, y3: 0.20 },
  { x: new Date(2004, 0, 1), y: 1.19, y1: 0.14, y2: 0.63, y3: 0.23 },
  { x: new Date(2005, 0, 1), y: 1.47, y1: 0.20, y2: 0.64, y3: 0.36 },
  { x: new Date(2006, 0, 1), y: 1.74, y1: 0.29, y2: 0.66, y3: 0.43 },
  { x: new Date(2007, 0, 1), y: 1.98, y1: 0.46, y2: 0.76, y3: 0.52 },
  { x: new Date(2008, 0, 1), y: 1.99, y1: 0.64, y2: 0.77, y3: 0.72 },
  { x: new Date(2009, 0, 1), y: 1.70, y1: 0.75, y2: 0.55, y3: 1.29 },
  { x: new Date(2010, 0, 1), y: 1.48, y1: 1.06, y2: 0.54, y3: 1.38 },
  { x: new Date(2011, 0, 1), y: 1.38, y1: 1.25, y2: 0.57, y3: 1.82 },
  { x: new Date(2012, 0, 1), y: 1.66, y1: 1.55, y2: 0.61, y3: 2.16 },
  { x: new Date(2013, 0, 1), y: 1.66, y1: 1.55, y2: 0.67, y3: 2.51 },
  { x: new Date(2014, 0, 1), y: 1.67, y1: 1.65, y2: 0.67, y3: 2.61 }
];

var chart = new ej.charts.Chart({
  primaryXAxis: {
    title: 'Years',
    valueType: 'DateTime',
    intervalType: 'Years',
    majorTickLines: { width: 0 },
    labelFormat: 'y',
    edgeLabelPlacement: 'Shift'
  },
  primaryYAxis: {
    {
      title: 'Spend in Billions',
      minimum: 0,
      maximum: 7,
      interval: 1,
      majorTickLines: { width: 0 },
      labelFormat: '{value}B'
    }
  },
  series: [
    {
      dataSource: chartData, xName: 'x', yName: 'y',
      //Series type as stacked area series
      type: 'StackingArea',
      name: 'Organic', border: { width: 2, color: 'black' },
      dashArray: '5.5'
    }, {
      dataSource: chartData, xName: 'x', yName: 'y1',
      type: 'StackingArea', name: 'Fair-trade',
      border: { width: 2, color: 'black' }, dashArray: '5.5'
    }, {
      dataSource: chartData, xName: 'x', yName: 'y2',
      type: 'StackingArea', name: 'Veg Alternatives',
      border: { width: 2, color: 'black' }, dashArray: '5.5'
    }, {
      dataSource: chartData, xName: 'x', yName: 'y3',
      type: 'StackingArea', name: 'Others',
      border: { width: 2, color: 'black' }, dashArray: '5.5'
    }
  ]
});

```



```
    ],
    title: 'Trend in Sales of Ethical Produce'
  }, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

See Also

- [Data label](#)
- [Tooltip](#)

Stacked area Chart in ##Platform_Name## control

100% Stacked Area

INDEX.JS

```
var chartData = [
  { x: new Date(2006, 0, 1), y: 34, y1: 51, y2: 14, y3: 37 },
  { x: new Date(2007, 0, 1), y: 20, y1: 26, y2: 34, y3: 15 },
  { x: new Date(2008, 0, 1), y: 40, y1: 37, y2: 73, y3: 53 },
  { x: new Date(2009, 0, 1), y: 51, y1: 51, y2: 51, y3: 51 },
  { x: new Date(2010, 0, 1), y: 26, y1: 26, y2: 26, y3: 26 },
  { x: new Date(2011, 0, 1), y: 37, y1: 37, y2: 37, y3: 37 },
  { x: new Date(2012, 0, 1), y: 54, y1: 43, y2: 12, y3: 54 },
  { x: new Date(2013, 0, 1), y: 44, y1: 23, y2: 16, y3: 44 },
  { x: new Date(2014, 0, 1), y: 48, y1: 55, y2: 34, y3: 23 }
];
```

```

var chart = new ej.charts.Chart({
  primaryXAxis: {
    title: 'Years',
    valueType: 'DateTime',
    intervalType: 'Years',
    labelFormat: 'y',
    edgeLabelPlacement: 'Shift'
  },
  primaryYAxis:
  {
    title: 'Temperature (%)',
    labelFormat: '{value}%',
    rangePadding: 'None'
  },
  series: [
    {
      dataSource: chartData, xName: 'x', yName: 'y',
      //Series type as 100% stacked area series
      type: 'StackingArea100',
      name: 'USA',
    }, {
      dataSource: chartData, xName: 'x', yName: 'y1',
      type: 'StackingArea100', name: 'UK',
    }, {
      dataSource: chartData, xName: 'x', yName: 'y2',
      type: 'StackingArea100', name: 'Canada',
    }, {
      dataSource: chartData, xName: 'x', yName: 'y3',
      type: 'StackingArea100', name: 'China',
    }
  ],
  title: 'Annual Temperature Comparison'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
</body>
</html>

```

```

var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Series customization

The following properties can be used to customize the 100% stacked area series.

- [fill](#) – Specifies the color of the area series.
- [opacity](#) – Specifies the opacity of [fill](#).
- [dashArray](#) – Specifies the dashes of series.

INDEX.JS

```

var chartData = [
    { x: new Date(2006, 0, 1), y: 34, y1: 51, y2: 14, y3: 37 },
    { x: new Date(2007, 0, 1), y: 20, y1: 26, y2: 34, y3: 15 },
    { x: new Date(2008, 0, 1), y: 40, y1: 37, y2: 73, y3: 53 },
    { x: new Date(2009, 0, 1), y: 51, y1: 51, y2: 51, y3: 51 },
    { x: new Date(2010, 0, 1), y: 26, y1: 26, y2: 26, y3: 26 },
    { x: new Date(2011, 0, 1), y: 37, y1: 37, y2: 37, y3: 37 },
    { x: new Date(2012, 0, 1), y: 54, y1: 43, y2: 12, y3: 54 },
    { x: new Date(2013, 0, 1), y: 44, y1: 23, y2: 16, y3: 44 },
    { x: new Date(2014, 0, 1), y: 48, y1: 55, y2: 34, y3: 23 }
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        title: 'Years',
        valueType: 'DateTime',
        intervalType: 'Years',
        labelFormat: 'y',
        edgeLabelPlacement: 'Shift'
    },
    primaryYAxis: {
        {
            title: 'Temperature (%)',
            labelFormat: '{value}%',
            rangePadding: 'None'
        }
    },
    series: [
        {
            dataSource: chartData, xName: 'x', yName: 'y',
            //Series type as 100% stacked area series
            type: 'StackingArea100', dashArray: '5.5',
            border: { width: 2, color: 'black' },
            name: 'USA',
        }, {
            dataSource: chartData, xName: 'x', yName: 'y1',
            type: 'StackingArea100', name: 'UK', dashArray: '5.5',
            border: { width: 2, color: 'black' }
        }, {
            dataSource: chartData, xName: 'x', yName: 'y2',

```

```

        type: 'StackingArea100', name: 'Canada', dashArray: '5.5',
        border: { width: 2, color: 'black' }
    }, {
        dataSource: chartData, xName: 'x', yName: 'y3',
        type: 'StackingArea100', name: 'China', dashArray: '5.5',
        border: { width: 2, color: 'black' }
    }
],
title: 'Annual Temperature Comparison'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Data label](#)
- [Tooltip](#)

Stacked step area Chart in ##Platform_Name## control

Stacked area

To render a stacked area series, use series [type](#) as `StackingArea` and inject `StackingAreaSeries` module using `Chart.Inject(StackingAreaSeries)` method.

INDEX.JS

```

var chartData = [

```

```

    { x: new Date(2000, 0, 1), y: 0.61, y1: 0.03, y2: 0.48, y3: 0.23 },
    { x: new Date(2001, 0, 1), y: 0.81, y1: 0.05, y2: 0.53, y3: 0.17 },
    { x: new Date(2002, 0, 1), y: 0.91, y1: 0.06, y2: 0.57, y3: 0.17 },
    { x: new Date(2003, 0, 1), y: 1, y1: 0.09, y2: 0.61, y3: 0.20 },
    { x: new Date(2004, 0, 1), y: 1.19, y1: 0.14, y2: 0.63, y3: 0.23 },
    { x: new Date(2005, 0, 1), y: 1.47, y1: 0.20, y2: 0.64, y3: 0.36 },
    { x: new Date(2006, 0, 1), y: 1.74, y1: 0.29, y2: 0.66, y3: 0.43 },
    { x: new Date(2007, 0, 1), y: 1.98, y1: 0.46, y2: 0.76, y3: 0.52 },
    { x: new Date(2008, 0, 1), y: 1.99, y1: 0.64, y2: 0.77, y3: 0.72 },
    { x: new Date(2009, 0, 1), y: 1.70, y1: 0.75, y2: 0.55, y3: 1.29 },
    { x: new Date(2010, 0, 1), y: 1.48, y1: 1.06, y2: 0.54, y3: 1.38 },
    { x: new Date(2011, 0, 1), y: 1.38, y1: 1.25, y2: 0.57, y3: 1.82 },
    { x: new Date(2012, 0, 1), y: 1.66, y1: 1.55, y2: 0.61, y3: 2.16 },
    { x: new Date(2013, 0, 1), y: 1.66, y1: 1.55, y2: 0.67, y3: 2.51 },
    { x: new Date(2014, 0, 1), y: 1.67, y1: 1.65, y2: 0.67, y3: 2.61 }
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        title: 'Years',
        valueType: 'DateTime',
        intervalType: 'Years',
        majorTickLines: { width: 0 },
        labelFormat: 'y',
        edgeLabelPlacement: 'Shift'
    },
    primaryYAxis: {
        title: 'Spend in Billions',
        minimum: 0,
        maximum: 7,
        interval: 1,
        majorTickLines: { width: 0 },
        labelFormat: '{value}B'
    },
    series: [
        {
            dataSource: chartData, xName: 'x', yName: 'y',
            //Series type as stacked area series
            type: 'StackingStepArea',
            name: 'Organic'
        }, {
            dataSource: chartData, xName: 'x', yName: 'y1',
            type: 'StackingStepArea', name: 'Fair-trade',
        }, {
            dataSource: chartData, xName: 'x', yName: 'y3',
            type: 'StackingStepArea', name: 'Others',
        }
    ],
    title: 'Trend in Sales of Ethical Produce'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">

```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Series customization

The following properties can be used to customize the **stacked step area** series.

- [fill](#) – Specifies the color of the area series.
- [opacity](#) – Specifies the opacity of [fill](#).
- [dashArray](#) – Specifies the dashes of series.
- [step](#) – Specifies the position of the step for the series.

INDEX.JS

```

var chartData = [
    { x: new Date(2000, 0, 1), y: 0.61, y1: 0.03, y2: 0.48, y3: 0.23 },
    { x: new Date(2001, 0, 1), y: 0.81, y1: 0.05, y2: 0.53, y3: 0.17 },
    { x: new Date(2002, 0, 1), y: 0.91, y1: 0.06, y2: 0.57, y3: 0.17 },
    { x: new Date(2003, 0, 1), y: 1, y1: 0.09, y2: 0.61, y3: 0.20 },
    { x: new Date(2004, 0, 1), y: 1.19, y1: 0.14, y2: 0.63, y3: 0.23 },
    { x: new Date(2005, 0, 1), y: 1.47, y1: 0.20, y2: 0.64, y3: 0.36 },
    { x: new Date(2006, 0, 1), y: 1.74, y1: 0.29, y2: 0.66, y3: 0.43 },
    { x: new Date(2007, 0, 1), y: 1.98, y1: 0.46, y2: 0.76, y3: 0.52 },
    { x: new Date(2008, 0, 1), y: 1.99, y1: 0.64, y2: 0.77, y3: 0.72 },
    { x: new Date(2009, 0, 1), y: 1.70, y1: 0.75, y2: 0.55, y3: 1.29 },
    { x: new Date(2010, 0, 1), y: 1.48, y1: 1.06, y2: 0.54, y3: 1.38 },
    { x: new Date(2011, 0, 1), y: 1.38, y1: 1.25, y2: 0.57, y3: 1.82 },
    { x: new Date(2012, 0, 1), y: 1.66, y1: 1.55, y2: 0.61, y3: 2.16 },
    { x: new Date(2013, 0, 1), y: 1.66, y1: 1.55, y2: 0.67, y3: 2.51 },
    { x: new Date(2014, 0, 1), y: 1.67, y1: 1.65, y2: 0.67, y3: 2.61 }
];
var chart = new ej.charts.Chart({
    primaryXAxis: {

```

```

        title: 'Years',
        valueType: 'DateTime',
        intervalType: 'Years',
        majorTickLines: { width: 0 },
        labelFormat: 'y',
        edgeLabelPlacement: 'Shift'
    },
    primaryYAxis:
    {
        title: 'Spend in Billions',
        minimum: 0,
        maximum: 7,
        interval: 1,
        majorTickLines: { width: 0 },
        labelFormat: '{value}B'
    },
    series: [
        {
            dataSource: chartData, xName: 'x', yName: 'y',
            //Series type as stacked area series
            type: 'StackingStepArea',
            name: 'Organic',border: { width:2, color: 'black'},
            dashArray: '5,5', fill:'red', opacity: 0.5, step: 'Center'
        }, {
            dataSource: chartData, xName: 'x', yName: 'y1',
            type: 'StackingStepArea', name: 'Fair-trade', opacity: 0.5,
            border: { width:2, color: 'black'}, dashArray: '5,5',
            fill:'green', step: 'Center'
        }, {
            dataSource: chartData, xName: 'x', yName: 'y3',
            type: 'StackingStepArea', name: 'Others',
            border: { width:2, color: 'black'}, dashArray: '5,5',
            fill:'blue',
            opacity: 0.5, step: 'Center'
        }
    ],
    title: 'Trend in Sales of Ethical Produce'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

```

```

<div id="container">
  <div id="element"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See also

- [Data label](#)
- [Tooltip](#)

Step area Chart in ##Platform_Name## control

Step area

To render a step area series, use series [type](#) as `StepArea` and inject `StepAreaSeries` module using `Chart.Inject(StepAreaSeries)` method.

INDEX.JS

```

var chartData = [
  { x: 1, y: 7 }, { x: 2, y: 1 }, { x: 3, y: 1 },
  { x: 4, y: 14 }, { x: 5, y: 1 }, { x: 6, y: 10 },
  { x: 7, y: 8 }, { x: 8, y: 6 }, { x: 9, y: 10 },
  { x: 10, y: 10 }, { x: 11, y: 16 }, { x: 12, y: 6 },
  { x: 13, y: 14 }, { x: 14, y: 7 }, { x: 15, y: 5 },
  { x: 16, y: 2 }, { x: 17, y: 14 }, { x: 18, y: 7 },
  { x: 19, y: 7 }, { x: 20, y: 10 }];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    valueType: 'Double',
    title: 'Overs'
  },
  primaryYAxis: {
    title: 'Runs'
  },
  series:[{
    dataSource: chartData,
    xName: 'x', yName: 'y',
    name: 'England', type: 'StepArea'
  }],
  title: 'England - Run Rate',
  tooltip:{
    enable:true
  }
}, '#element');

```

INDEX.HTML


```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Series customization

The following properties can be used to customize the **step area** series.

- [fill](#) – Specifies the color of the area series.
- [opacity](#) – Specifies the opacity of [fill](#).
- [dashArray](#) – Specifies the dashes of series.
- [step](#) – Specifies the position of the step for the series.

INDEX.JS

```

var chartData = [
  { x: 1, y: 7 }, { x: 2, y: 1 }, { x: 3, y: 1 },
  { x: 4, y: 14 }, { x: 5, y: 1 }, { x: 6, y: 10 },
  { x: 7, y: 8 }, { x: 8, y: 6 }, { x: 9, y: 10 },
  { x: 10, y: 10 }, { x: 11, y: 16 }, { x: 12, y: 6 },
  { x: 13, y: 14 }, { x: 14, y: 7 }, { x: 15, y: 5 },
  { x: 16, y: 2 }, { x: 17, y: 14 }, { x: 18, y: 7 },
  { x: 19, y: 7 }, { x: 20, y: 10 }];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    valueType: 'Double',
    title: 'Overs'
  },
  primaryYAxis: {
    title: 'Runs'
  },
});

```

```

series:[{
  dataSource: chartData,
  xName: 'x', yName: 'y',
  name: 'England', type: 'StepArea', border: { width:2, color:
'red'},
  opacity: 0.4, step: 'Right', dashArray: '5,5'
}],
title: 'England - Run Rate',
tooltip:{
  enable:true
}
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

See also

- [Data label](#)
- [Tooltip](#)

Spline area Chart in ##Platform_Name## control

Spline area

To render a spline area series, use series [type](#) as `SplineArea` and inject `SplineAreaSeries` module using `Chart.Inject(SplineAreaSeries)` method.

INDEX.JS

```

var chartData = [
    { x: 'Jan', y: -1 }, { x: 'Feb', y: -1 }, { x: 'Mar', y: 2 },
    { x: 'Apr', y: 8 }, { x: 'May', y: 13 }, { x: 'Jun', y: 18 },
    { x: 'Jul', y: 21 }, { x: 'Aug', y: 20 }, { x: 'Sep', y: 16 },
    { x: 'Oct', y: 10 }, { x: 'Nov', y: 4 }, { x: 'Dec', y: 0 }
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        title: 'Month',
        valueType: 'Category'
    },
    primaryYAxis: {
        minimum: -5, maximum: 35, interval: 5,
        title: 'Temperature in Celsius',
        labelFormat: '{value}C'
    },
    series:[{
        dataSource: chartData, width:2,
        xName: 'x', yName: 'y',
        name: 'London',
        // Series type as spline series
        type: 'Spline'
    }],
    title: 'Climate Graph-2012'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>

```

```
</body></html>
```

Series customization

The following properties can be used to customize the **spline area** series.

- [fill](#) – Specifies the color of the area series.
- [opacity](#) – Specifies the opacity of [fill](#).
- [dashArray](#) – Specifies the dashes of series.

INDEX.JS

```
var chartData = [
  { x: 1, y: 7 }, { x: 2, y: 5 }, { x: 3, y: 1 },
  { x: 4, y: 14 }, { x: 5, y: 1 }, { x: 6, y: 10 },
  { x: 7, y: 8 }, { x: 8, y: 6 }, { x: 9, y: 10 },
  { x: 10, y: 10 }, { x: 11, y: 16 }, { x: 12, y: 6 },
  { x: 13, y: 14 }, { x: 14, y: 7 }, { x: 15, y: 5 },
  { x: 16, y: 2 }, { x: 17, y: 14 }, { x: 18, y: 7 },
  { x: 19, y: 7 }, { x: 20, y: 10 }];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    valueType: 'Double',
    title: 'Overs'
  },
  primaryYAxis: {
    title: 'Runs'
  },
  series:[{
    dataSource: chartData,
    xName: 'x', yName: 'y',
    name: 'England', type: 'SplineArea', border: { width:2, color:
'red' }
  }],
  title: 'England - Run Rate',
  tooltip:{
    enable:true
  }
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
```

```

</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Data label](#)
- [Tooltip](#)

Column Chart in ##Platform_Name## control

Column

INDEX.JS

```

var chartData = [
    { country: "USA", gold: 50 },
    { country: "China", gold: 40 },
    { country: "Japan", gold: 70 },
    { country: "Australia", gold: 60 },
    { country: "France", gold: 50 },
    { country: "Germany", gold: 40 },
    { country: "Italy", gold: 40 },
    { country: "Sweden", gold: 30 }
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        valueType: 'Category',
        title: 'Countries'
    },
    primaryYAxis: {
        minimum: 0, maximum: 80,
        interval: 20, title: 'Medals'
    },
    series:[{
        dataSource: chartData,
        xName: 'country', yName: 'gold',
        name: 'Gold',
        // Series type as column series
        type: 'Column'
    }],
    title: 'Olympic Medals'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Column space and width

The [columnSpacing](#) and [columnWidth](#) properties are used to customize the space between columns.

INDEX.JS

```

var chartData = [
  { country: 'USA', gold: 50, silver: 40 },
  { country: 'China', gold: 40, silver: 35 },
  { country: 'Japan', gold: 70, silver: 65 },
  { country: 'Australia', gold: 60, silver: 50 },
  { country: 'France', gold: 50, silver: 55 },
  { country: 'Germany', gold: 40, silver: 20 },
  { country: 'Italy', gold: 40, silver: 30 },
  { country: 'Sweden', gold: 30, silver: 40 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    valueType: 'Category',
    title: 'Countries'
  },
  primaryYAxis: {
    minimum: 0, maximum: 80,
    interval: 20, title: 'Medals'
  },
  series:[{
    dataSource: chartData,
    xName: 'country', yName: 'gold',
    name: 'Gold',

```

```

        // Series type as column series
        type: 'Column'
    },
    {
        dataSource: chartData,
        xName: 'country',
        yName: 'silver',
        name: 'Silver',
        columnWidth: 0.75,
        columnSpacing: 1.5,
        // Series type as column series
        type: 'Column',
    }],
    title: 'Olympic Medals'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Grouped column

You can use the [groupName](#) property to group the data points in the column type charts. Data points with same group name are grouped together.

INDEX.JS

```

var chart = new ej.charts.Chart({
  primaryXAxis: {
    valueType: 'Category',

```

```

    },
    series: [
        {
            type: 'Column', xName: 'x', width: 2, yName: 'y', groupName:
            'USA', columnWidth: 0.7,
            dataSource: [{ x: '2012', y: 104 }, { x: '2016', y: 121 }, {
            x: '2020', y: 113 }], columnSpacing: 0.1,
        },
        {
            type: 'Column', xName: 'x', width: 2, yName: 'y', groupName:
            'USA', columnWidth: 0.5,
            dataSource: [{ x: '2012', y: 46 }, { x: '2016', y: 46 }, {
            x: '2020', y: 39 }], columnSpacing: 0.1,
        },
        {
            type: 'Column', xName: 'x', width: 2, yName: 'y', groupName:
            'UK', columnWidth: 0.7,
            dataSource: [{ x: '2012', y: 65 }, { x: '2016', y: 67 }, { x:
            '2020', y: 65 }], columnSpacing: 0.1,
        },
        {
            type: 'Column', xName: 'x', width: 2, yName: 'y', groupName:
            'UK', columnWidth: 0.5,
            dataSource: [{ x: '2012', y: 29 }, { x: '2016', y: 27 }, { x:
            '2020', y: 22 }], columnSpacing: 0.1,
        },
    ],
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');

```



```

if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Cylindrical column chart

To render a cylindrical column chart, set the [columnFacet](#) property to **Cylinder** in the chart series.

INDEX.JS

```

var cylindricalData = [
    { country: "USA", gold: 50, tooltipMappingName: 'USA' },
    { country: "Japan", gold: 70, tooltipMappingName: 'Japan' },
    { country: "Australia", gold: 60, tooltipMappingName: 'Australia' },
    { country: "France", gold: 50, tooltipMappingName: 'France' },
    { country: "Italy", gold: 40, tooltipMappingName: 'Italy' },
    { country: "Sweden", gold: 55, tooltipMappingName: 'Sweden' }
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        valueType: 'Category',
        interval: 1
    },
    primaryYAxis: {
        minimum: 0, maximum: 80,
        interval: 10, title: 'Medal Count'
    },
    series: [{
        dataSource: cylindricalData,
        xName: 'country', yName: 'gold',
        tooltipMappingName: 'tooltipMappingName',
        // Series type as column series with cylinder shape
        type: 'Column', columnFacet: 'Cylinder'
    }],
    title: 'Olympic Gold Medal Counts - RIO',
    tooltip: { enable: true, header: "<b>${point.tooltip}</b>", format:
    "Gold Medal: <b>${point.y}</b>" }
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>

```

```

</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Series customization

The following properties can be used to customize the **column** series.

- [fill](#) – Specifies the color of the series.
- [opacity](#) – Specifies the opacity of [fill](#).
- [dashArray](#) – Specifies the dashes of series.
- [border](#) – Specifies the [color](#) and [width](#) of series border.

INDEX.JS

```

var chartData = [
    { country: "USA", gold: 50 },
    { country: "China", gold: 40 },
    { country: "Japan", gold: 70 },
    { country: "Australia", gold: 60 },
    { country: "France", gold: 50 },
    { country: "Germany", gold: 40 },
    { country: "Italy", gold: 40 },
    { country: "Sweden", gold: 30 }
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        valueType: 'Category',
        title: 'Countries'
    },
    primaryYAxis: {
        minimum: 0, maximum: 80,
        interval: 20, title: 'Medals'
    },
    series:[{
        dataSource: chartData,
        xName: 'country', yName: 'gold',
        //fill for column series
        fill: 'green',
        //border width and color for column series
        border: {
            width: '2',
            color: 'red'
        },
    },

```

```

        columnSpacing: 0.5,
        columnWidth: 0.5,
        name: 'Gold',
        //series type as column series
        type: 'Column'
    }],
    title: 'Olympic Medals'
}, '#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

See also

- [Data label](#)
- [Tooltip](#)

Range column Chart in ##Platform_Name## control

Range Column

To render a range column series, use series [type](#) as `RangeColumn` and inject `RangeColumnSeries` module using `Chart.Inject(RangeColumnSeries)` method.

INDEX.JS

```

var data = [
  { x: 'Jan', low: 0.7, high: 6.1 }, { x: 'Feb', low: 1.3, high: 6.3 }, {
  x: 'Mar', low: 1.9, high: 8.5 },
```

```

    { x: 'Apr', low: 3.1, high: 10.8 }, { x: 'May', low: 5.7, high: 14.40 },
    { x: 'Jun', low: 8.4, high: 16.90 },
    { x: 'Jul', low: 10.6, high: 19.20 }, { x: 'Aug', low: 10.5, high: 18.9 },
    { x: 'Sep', low: 8.5, high: 16.1 },
    { x: 'Oct', low: 6.0, high: 12.5 }, { x: 'Nov', low: 1.5, high: 6.9 },
    { x: 'Dec', low: 5.1, high: 12.1 }
  ];
  var data2 = [
    { x: 'Jan', low: 1.7, high: 7.1 }, { x: 'Feb', low: 1.9, high: 7.7 }, {
  x: 'Mar', low: 1.2, high: 7.5 },
    { x: 'Apr', low: 2.5, high: 9.8 }, { x: 'May', low: 4.7, high: 11.4 }, {
  x: 'Jun', low: 6.4, high: 14.4 },
    { x: 'Jul', low: 9.6, high: 17.2 }, { x: 'Aug', low: 10.7, high: 17.9 },
    { x: 'Sep', low: 7.5, high: 15.1 },
    { x: 'Oct', low: 3.0, high: 10.5 }, { x: 'Nov', low: 1.2, high: 7.9 }, {
  x: 'Dec', low: 4.1, high: 9.1 }
  ];
  var chart = new ej.charts.Chart({
    primaryXAxis: {
      title: 'Month',
      valueType: 'Category'
    },
    primaryYAxis:
    {
      title: 'Temperature(Celsius)',
    },
    series: [
      {
        // Series type as range column series
        type: 'RangeColumn',
        name: 'India',
        dataSource: data, xName: 'x', high: 'high', low: 'low',
      }, {
        type: 'RangeColumn', name: 'India',
        dataSource: data2, xName: 'x', high: 'high', low: 'low',
      }
    ],
    title: 'Maximum and Minimum Temperature'
  }, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>

```

```

<body>

    <div id="container">
        <div id="element"></div>
    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Series customization

The following properties can be used to customize the **range column** series.

- [fill](#) – Specifies the color of the series.
- [opacity](#) – Specifies the opacity of [fill](#).
- [dashArray](#) – Specifies the dashes of series.
- [border](#) – Specifies the [color](#) and [width](#) of series border.

INDEX.JS

```

var data = [
    { x: 'Jan', low: 0.7, high: 6.1 }, { x: 'Feb', low: 1.3, high: 6.3 }, {
x: 'Mar', low: 1.9, high: 8.5 },
    { x: 'Apr', low: 3.1, high: 10.8 }, { x: 'May', low: 5.7, high: 14.40 },
    { x: 'Jun', low: 8.4, high: 16.90 },
    { x: 'Jul', low: 10.6, high: 19.20 }, { x: 'Aug', low: 10.5, high: 18.9 },
    { x: 'Sep', low: 8.5, high: 16.1 },
    { x: 'Oct', low: 6.0, high: 12.5 }, { x: 'Nov', low: 1.5, high: 6.9 },
    { x: 'Dec', low: 5.1, high: 12.1 }
];
var data2 = [
    { x: 'Jan', low: 1.7, high: 7.1 }, { x: 'Feb', low: 1.9, high: 7.7 }, {
x: 'Mar', low: 1.2, high: 7.5 },
    { x: 'Apr', low: 2.5, high: 9.8 }, { x: 'May', low: 4.7, high: 11.4 }, {
x: 'Jun', low: 6.4, high: 14.4 },
    { x: 'Jul', low: 9.6, high: 17.2 }, { x: 'Aug', low: 10.7, high: 17.9 },
    { x: 'Sep', low: 7.5, high: 15.1 },
    { x: 'Oct', low: 3.0, high: 10.5 }, { x: 'Nov', low: 1.2, high: 7.9 }, {
x: 'Dec', low: 4.1, high: 9.1 }
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        title: 'Month',
        valueType: 'Category'
    },
    primaryYAxis:
    {
        title: 'Temperature(Celsius)',
    },
    series: [

```

```

    {
        // Series type as range column series
        type: 'RangeColumn',
        name: 'India', fill: 'red', border: { width: 2, color: 'green' },
        dataSource: data, xName: 'x', high: 'high', low: 'low',
    }, {
        type: 'RangeColumn', name: 'India', fill: 'red', border: { width:
2, color: 'green' },
        dataSource: data2, xName: 'x', high: 'high', low: 'low',
    }
],
title: 'Maximum and Minimum Temperature'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Data label](#)
- [Tooltip](#)

Stack column Chart in ##Platform_Name## control

Stacked column

To render a stacked column series, use series [type](#) as `StackingColumn` and inject `StackingColumnSeries` module using `Chart.Inject(StackingColumnSeries)` method.

INDEX.JS

```

var chartData = [
  { x: '2014', y: 111.1, y1: 76.9, y2: 66.1, y3: 34.1 },
  { x: '2015', y: 127.3, y1: 99.5, y2: 79.3, y3: 38.2 },
  { x: '2016', y: 143.4, y1: 121.7, y2: 91.3, y3: 44.0 },
  { x: '2017', y: 159.9, y1: 142.5, y2: 102.4, y3: 51.6 },
  { x: '2018', y: 175.4, y1: 166.7, y2: 112.9, y3: 61.9 },
  { x: '2019', y: 189.0, y1: 182.9, y2: 122.4, y3: 71.5 },
  { x: '2020', y: 202.7, y1: 197.3, y2: 120.9, y3: 82.0 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    title: 'Years',
    interval: 1,
    valueType: 'Category'
  },
  primaryYAxis: {
    title: 'Sales in Billions',
    minimum: 0,
    maximum: 700,
    interval: 100,
    labelFormat: '{value}B',
  },
  series: [
    {
      dataSource: chartData, xName: 'x', yName: 'y',
      //Series type as stacked column
      type: 'StackingColumn',
      name: 'UK',
    }, {
      dataSource: chartData, xName: 'x', yName: 'y1',
      type: 'StackingColumn', name: 'Germany',
    }, {
      dataSource: chartData, xName: 'x', yName: 'y2',
      type: 'StackingColumn', name: 'France',
    }, {
      dataSource: chartData, xName: 'x', yName: 'y3',
      type: 'StackingColumn', name: 'Italy',
    }
  ],
  title: 'Mobile Game Market by Country'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Stacking group

You can use the [stackingGroup](#) property to group the stacked columns and 100% stacked columns.

Columns with same group name are stacked on top of each other.

INDEX.JS

```

var chartData = [
    { x: '2014', y: 111.1, y1: 76.9, y2: 66.1, y3: 34.1 },
    { x: '2015', y: 127.3, y1: 99.5, y2: 79.3, y3: 38.2 },
    { x: '2016', y: 143.4, y1: 121.7, y2: 91.3, y3: 44.0 },
    { x: '2017', y: 159.9, y1: 142.5, y2: 102.4, y3: 51.6 },
    { x: '2018', y: 175.4, y1: 166.7, y2: 112.9, y3: 61.9 },
    { x: '2019', y: 189.0, y1: 182.9, y2: 122.4, y3: 71.5 },
    { x: '2020', y: 202.7, y1: 197.3, y2: 120.9, y3: 82.0 }
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        title: 'Years',
        interval: 1,
        valueType: 'Category'
    },
    primaryYAxis: {
        title: 'Sales in Billions',
        minimum: 0,
        maximum: 400,
        interval: 100,
        labelFormat: '{value}B',
    },
    series: [
        {
            dataSource: chartData, xName: 'x', yName: 'y',
            type: 'StackingColumn', name: 'UK',
            //Stacking group for stacked column series
            stackingGroup: 'UKAndGermany'
        }, {

```



```

        dataSource: chartData, xName: 'x', yName: 'y1',
        type: 'StackingColumn', name: 'Germany', stackingGroup:
'UKAndGermany'
    }, {
        dataSource: chartData, xName: 'x', yName: 'y2',
        type: 'StackingColumn', name: 'France', stackingGroup:
'FranceAndItaly'
    }, {
        dataSource: chartData, xName: 'x', yName: 'y3',
        type: 'StackingColumn', name: 'Italy', stackingGroup:
'FranceAndItaly'
    }
],
    title: 'Mobile Game Market by Country'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Cylindrical stacked column chart

To render a cylindrical stacked column chart, set the [columnFacet](#) property to `Cylinder` in the chart series.

INDEX.JS

```

var cylindricalData = [
  { x: '2014', y: 111.1, y1: 76.9, y2: 66.1, y3: 34.1 },
  { x: '2015', y: 127.3, y1: 99.5, y2: 79.3, y3: 38.2 },

```

```

{ x: '2016', y: 143.4, y1: 121.7, y2: 91.3, y3: 44.0 },
{ x: '2017', y: 159.9, y1: 142.5, y2: 102.4, y3: 51.6 },
{ x: '2018', y: 175.4, y1: 166.7, y2: 112.9, y3: 61.9 },
{ x: '2019', y: 189.0, y1: 182.9, y2: 122.4, y3: 71.5 },
{ x: '2020', y: 202.7, y1: 197.3, y2: 120.9, y3: 82.0 }];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    title: 'Years',
    interval: 1,
    valueType: 'Category'
  },
  primaryYAxis:
  {
    title: 'Sales in Billions',
    minimum: 0,
    maximum: 700,
    interval: 100,
    labelFormat: '{value}B',
  },
  //Series type as stacked column with cylinder shape
  series: [
    {
      dataSource: cylindricalData, xName: 'x', yName: 'y',
      type: 'StackingColumn', columnFacet: 'Cylinder', name: 'UK'
    },
    {
      dataSource: cylindricalData, xName: 'x', yName: 'y1',
      type: 'StackingColumn', columnFacet: 'Cylinder', name: 'Germany'
    },
    {
      dataSource: cylindricalData, xName: 'x', yName: 'y2',
      type: 'StackingColumn', columnFacet: 'Cylinder', name: 'France'
    },
    {
      dataSource: cylindricalData, xName: 'x', yName: 'y3',
      type: 'StackingColumn', columnFacet: 'Cylinder', name: 'Italy'
    }
  ],
  title: 'Mobile Game Market by Country'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>

```

```

</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Series customization

The following properties can be used to customize the **column** series.

- [fill](#) – Specifies the color of the series.
- [opacity](#) – Specifies the opacity of [fill](#).
- [dashArray](#) – Specifies the dashes of series.
- [border](#) – Specifies the [color](#) and [width](#) of series border.

INDEX.JS

```

var chartData = [
    { x: '2014', y: 111.1, y1: 76.9, y2: 66.1, y3: 34.1 },
    { x: '2015', y: 127.3, y1: 99.5, y2: 79.3, y3: 38.2 },
    { x: '2016', y: 143.4, y1: 121.7, y2: 91.3, y3: 44.0 },
    { x: '2017', y: 159.9, y1: 142.5, y2: 102.4, y3: 51.6 },
    { x: '2018', y: 175.4, y1: 166.7, y2: 112.9, y3: 61.9 },
    { x: '2019', y: 189.0, y1: 182.9, y2: 122.4, y3: 71.5 },
    { x: '2020', y: 202.7, y1: 197.3, y2: 120.9, y3: 82.0 }
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        title: 'Years',
        interval: 1,
        valueType: 'Category'
    },
    primaryYAxis: {
        {
            title: 'Sales in Billions',
            minimum: 0,
            maximum: 400,
            interval: 100,
            labelFormat: '{value}B',
        },
        series: [
            {
                dataSource: chartData, xName: 'x', yName: 'y1', border: {
width: 1.5, color: 'blue'},
                type: 'StackingColumn', stackingGroup: 'UKAndGermany'
            }, {

```

```

        dataSource: chartData, xName: 'x', yName: 'y2',
        type: 'StackingColumn', stackingGroup:
'FranceAndItaly',border: { width: 1.5, color: 'yellow'}
    }, {
        dataSource: chartData, xName: 'x', yName: 'y3',
        type: 'StackingColumn', stackingGroup:
'FranceAndItaly',border: { width: 1.5, color: 'red'}
    }
],
title: 'Mobile Game Market by Country'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

See also

- [Data label](#)
- [Tooltip](#)

Stacked column Chart in ##Platform_Name## control

100% Stacked column

INDEX.JS

```

var chartData = [
  { x: new Date(2006, 0, 1), y: 900, y1: 190, y2: 250, y3: 150 },
  { x: new Date(2007, 0, 1), y: 544, y1: 226, y2: 145, y3: 120 },

```

```

    { x: new Date(2008, 0, 1), y: 880, y1: 194, y2: 190, y3: 115 },
    { x: new Date(2009, 0, 1), y: 675, y1: 250, y2: 220, y3: 125 },
    { x: new Date(2010, 0, 1), y: 765, y1: 222, y2: 225, y3: 132 },
    { x: new Date(2011, 0, 1), y: 679, y1: 181, y2: 135, y3: 137 },
    { x: new Date(2012, 0, 1), y: 770, y1: 128, y2: 152, y3: 110 },
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        title: 'Years',
        interval: 1,
        valueType: 'DateTime',
        labelFormat: 'y'
    },
    primaryYAxis:
    {
        title: 'GDP (%) Per Annum',
        rangePadding: 'None',
        labelFormat: '{value}%',
    },
    series: [
        {
            dataSource: chartData, xName: 'x', yName: 'y',
            //Series type as 100% stacked column series
            type: 'StackingColumn100',
            name: 'UK',
        }, {
            dataSource: chartData, xName: 'x', yName: 'y1',
            type: 'StackingColumn100', name: 'Germany',
        }, {
            dataSource: chartData, xName: 'x', yName: 'y2',
            type: 'StackingColumn100', name: 'France',
        }, {
            dataSource: chartData, xName: 'x', yName: 'y3',
            type: 'StackingColumn100', name: 'Italy',
        }
    ],
    title: 'Gross Domestic Product Growth'
}, '#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>
```

```

    <div id="container">
      <div id="element"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

100% Cylindrical stacked column chart

To render a 100% cylindrical stacked column chart, set the [columnFacet](#) property to **Cylinder** in the chart series.

INDEX.JS

```

var cylindricalData = [
  { x: new Date(2006, 0, 1), y: 900, y1: 190, y2: 250, y3: 150 },
  { x: new Date(2007, 0, 1), y: 544, y1: 226, y2: 145, y3: 120 },
  { x: new Date(2008, 0, 1), y: 880, y1: 194, y2: 190, y3: 115 },
  { x: new Date(2009, 0, 1), y: 675, y1: 250, y2: 220, y3: 125 },
  { x: new Date(2010, 0, 1), y: 765, y1: 222, y2: 225, y3: 132 },
  { x: new Date(2011, 0, 1), y: 679, y1: 181, y2: 135, y3: 137 },
  { x: new Date(2012, 0, 1), y: 770, y1: 128, y2: 152, y3: 110 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    title: 'Years',
    interval: 1,
    valueType: 'DateTime',
    labelFormat: 'y'
  },
  primaryYAxis: {
    {
      title: 'GDP (%) Per Annum',
      rangePadding: 'None',
      labelFormat: '{value}%',
    },
    //Series type as 100% stacked column series with cylindrical shape
    series: [
      {
        dataSource: cylindricalData, xName: 'x', yName: 'y',
        type: 'StackingColumn100', columnFacet: 'Cylinder', name: 'UK'
      },
      {
        dataSource: cylindricalData, xName: 'x', yName: 'y1',
        type: 'StackingColumn100', columnFacet: 'Cylinder', name:
'Germany'
      },
      {
        dataSource: cylindricalData, xName: 'x', yName: 'y2',
        type: 'StackingColumn100', columnFacet: 'Cylinder', name:
'France'
      }
    ]
  }
});

```

```

    },
    {
        dataSource: cylindricalData, xName: 'x', yName: 'y3',
        type: 'StackingColumn100', columnFacet: 'Cylinder', name:
'Italy'
    }
],
title: 'Gross Domestic Product Growth'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Series customization

The following properties can be used to customize the **column** series.

- [fill](#) – Specifies the color of the series.
- [opacity](#) – Specifies the opacity of [fill](#).
- [dashArray](#) – Specifies the dashes of series.
- [border](#) – Specifies the [color](#) and [width](#) of series border.

INDEX.JS

```

var chartData = [
  { x: new Date(2006, 0, 1), y: 900, y1: 190, y2: 250, y3: 150 },
  { x: new Date(2007, 0, 1), y: 544, y1: 226, y2: 145, y3: 120 },
  { x: new Date(2008, 0, 1), y: 880, y1: 194, y2: 190, y3: 115 },

```

```

    { x: new Date(2009, 0, 1), y: 675, y1: 250, y2: 220, y3: 125 },
    { x: new Date(2010, 0, 1), y: 765, y1: 222, y2: 225, y3: 132 },
    { x: new Date(2011, 0, 1), y: 679, y1: 181, y2: 135, y3: 137 },
    { x: new Date(2012, 0, 1), y: 770, y1: 128, y2: 152, y3: 110 },
  ];
  var chart = new ej.charts.Chart({
    primaryXAxis: {
      title: 'Years',
      interval: 1,
      valueType: 'DateTime',
      labelFormat: 'y'
    },
    primaryYAxis:
    {
      title: 'GDP (%) Per Annum',
      rangePadding: 'None',
      labelFormat: '{value}%',
    },
    series: [
      {
        dataSource: chartData, xName: 'x', yName: 'y1',
        type: 'StackingColumn100', name: 'Germany',border: { width:
1.5, color: 'red' }
      }, {
        dataSource: chartData, xName: 'x', yName: 'y2',
        type: 'StackingColumn100', name: 'France', border: { width:
1.5, color: 'yellow' }
      }, {
        dataSource: chartData, xName: 'x', yName: 'y3',
        type: 'StackingColumn100', name: 'Italy', border: { width:
1.5, color: 'blue' }
      }
    ],
    title: 'Gross Domestic Product Growth'
  }, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>

```



```

    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See also

- [Data label](#)
- [Tooltip](#)

Bar Chart in ##Platform_Name## control

Bar

INDEX.JS

```

var chartData = [
    { x: 2006, y: 7.8 }, { x: 2007, y: 7.2 },
    { x: 2008, y: 6.8 }, { x: 2009, y: 10.7 },
    { x: 2010, y: 10.8 }, { x: 2011, y: 9.8 }
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        minimum: 2005, maximum: 2012, interval: 1,
        title: 'Year'
    },
    primaryYAxis: {
        minimum: 3, maximum: 12,
        interval: 1, title: 'Percentage',
        labelFormat: '{value}%'
    },
    series:[{
        dataSource: chartData,
        xName: 'x', yName: 'y',
        name: 'India',
        // Series type as bar series
        type: 'Bar'
    }],
    title: 'Unemployment rate (%)'
}, '#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Bar space and width

The [columnSpacing](#) and [columnWidth](#) properties are used to customize the space between bars.

INDEX.JS

```

var chartData = [
    { x: 2005, y: 8, y1: 4 },
    { x: 2006, y: 5, y1: 8 },
    { x: 2007, y: 6, y1: 3.5 },
    { x: 2008, y: 7, y1: 6 },
    { x: 2009, y: 3.5, y1: 4 },
    { x: 2010, y: 5, y1: 3.5 },
    { x: 2011, y: 3.5, y1: 5 }
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        minimum: 2005, maximum: 2012, interval: 1,
        title: 'Year'
    },
    primaryYAxis: {
        minimum: 3, maximum: 12,
        interval: 1, title: 'Percentage',
        labelFormat: '{value}%'
    },
    series: [{
        dataSource: chartData,
        xName: 'x', yName: 'y',
        name: 'India',
        // Series type as bar series
        type: 'Bar'
    },
    {
        dataSource: chartData,
        xName: 'x', yName: 'y1',
        name: 'India',
        columnSpacing: 0.5,

```

```

        columnWidth: 0.75,
        // Series type as bar series
        type: 'Bar'
    }],
    title: 'Unemployment rate (%)'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Grouped bar

You can use the [groupName](#) property to group the data points in the bar type charts. Data points with same group name are grouped together.

INDEX.JS

```

var chart = new ej.charts.Chart({
  primaryXAxis: {
    valueType: 'Category',
  },
  series: [
    {
      type: 'Bar', xName: 'x', width: 2, yName: 'y', groupName:
      'USA', columnWidth: 0.7,
      dataSource: [{ x: '2012', y: 104 }, { x: '2016', y: 121 }, {
      x: '2020', y: 113 }], columnSpacing: 0.1,
    },
  ],
});

```

```

        {
            type: 'Bar', xName: 'x', width: 2, yName: 'y', groupName:
'USA', columnWidth: 0.5,
            dataSource: [{ x: '2012', y: 46 }, { x: '2016', y: 46 }, {
x: '2020', y: 39 }], columnSpacing: 0.1,
        },
        {
            type: 'Bar', xName: 'x', width: 2, yName: 'y', groupName:
'UK', columnWidth: 0.7,
            dataSource: [{ x: '2012', y: 65 }, { x: '2016', y: 67 }, { x:
'2020', y: 65 }], columnSpacing: 0.1,
        },
        {
            type: 'Bar', xName: 'x', width: 2, yName: 'y', groupName:
'UK', columnWidth: 0.5,
            dataSource: [{ x: '2012', y: 29 }, { x: '2016', y: 27 }, { x:
'2020', y: 22 }], columnSpacing: 0.1,
        },
    ],
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Cylindrical bar chart

To render a cylindrical bar chart, set the [columnFacet](#) property to **Cylinder** in the chart series.

INDEX.JS

```
var cylindricalData = [
  { x: 2006, y: 9 }, { x: 2007, y: 7.8 },
  { x: 2008, y: 10.5 }, { x: 2009, y: 8.4 },
  { x: 2010, y: 6 }, { x: 2011, y: 11 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    minimum: 2005, maximum: 2012, interval: 1
  },
  primaryYAxis: {
    minimum: 3, maximum: 12,
    interval: 1, title: 'Percentage'
  },
  series: [{
    dataSource: cylindricalData,
    xName: 'x', yName: 'y',
    // Series type as bar series with cylinder shape
    type: 'Bar', columnFacet: 'Cylinder'
  }],
  title: 'Unemployment rate in percentage'
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Series customization

The following properties can be used to customize the **bar** series.

- [fill](#) – Specifies the color of the series.
- [opacity](#) – Specifies the opacity of [fill](#).
- [dashArray](#) – Specifies the dashes of series.
- [border](#) – Specifies the [color](#) and [width](#) of series border.

INDEX.JS

```
var chartData = [
  { x: 2006, y: 7.8 }, { x: 2007, y: 7.2 },
  { x: 2008, y: 6.8 }, { x: 2009, y: 10.7 },
  { x: 2010, y: 10.8 }, { x: 2011, y: 9.8 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    minimum: 2005, maximum: 2012, interval: 1,
    title: 'Year'
  },
  primaryYAxis: {
    minimum: 3, maximum: 12,
    interval: 1, title: 'Percentage',
    labelFormat: '{value}%'
  },
  series:[{
    dataSource: chartData,
    xName: 'x', yName: 'y',
    //fill for bar series
    fill: 'green',
    //border for bar series
    border: {
      width: 2,
      color: 'red'
    },
    columnWidth: 0.5,
    columnSpacing: 0.5,
    name: 'India',
    // series type as bar series
    type: 'Bar'
  }],
  title: 'Unemployment rate (%) '
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See also

- [Data label](#)
- [Tooltip](#)

Stack bar Chart in ##Platform_Name## control

Stacked bar

To render a stacked bar series, use series [type](#) as `StackingBar` and inject `StackingBarSeries` module using `Chart.Inject(StackingBarSeries)` method.

INDEX.JS

```

var chartData = [
    { x: 'Jan', y: 6, y1: 6, y2: -1 }, { x: 'Feb', y: 8, y1: 8, y2: -1.5 },
    { x: 'Mar', y: 12, y1: 11, y2: -2 }, { x: 'Apr', y: 15, y1: 16, y2: -2.5 },
    { x: 'May', y: 20, y1: 21, y2: -3 }, { x: 'Jun', y: 24, y1: 25, y2: -3.5 },
    { x: 'Jul', y: 28, y1: 27, y2: -4 }, { x: 'Aug', y: 32, y1: 31, y2: -4.5 },
    { x: 'Sep', y: 33, y1: 34, y2: -5 }, { x: 'Oct', y: 35, y1: 34, y2: -5.5 },
    { x: 'Nov', y: 40, y1: 41, y2: -6 }, { x: 'Dec', y: 42, y1: 42, y2: -6.5 },
    { x: 'Dec', y: 42, y1: 42, y2: -6.5 }
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        valueType: 'Category',
        title: 'Months'
    },
    primaryYAxis: {
        title: 'Percentage (%)',

```

```

        minimum: -20,
        maximum: 100,
        labelFormat: '{value}%',
        edgeLabelPlacement: 'Shift'
    },
    series: [
        {
            //Series type as stacked bar
            type: 'StackingBar', name: 'Apple',
            dataSource: chartData, xName: 'x', yName: 'y'
        }, {
            type: 'StackingBar', name: 'Orange',
            dataSource: chartData, xName: 'x', yName: 'y1'
        }, {
            type: 'StackingBar', name: 'Wastage',
            dataSource: chartData, xName: 'x', yName: 'y2'
        }
    ],
    title: 'Sales Comparison'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Stacking group

You can use the [stackingGroup](#) property to group the stacked bar and 100% stacked bar. Columns with same group name are stacked on top of each other.

INDEX.JS

```

var chartData = [
  { x: 2007, y: 453, y1: 876, y2: 356, y3: 122 }, { x: 2008, y: 354, y1: 564, y2: 876, y3: 444 },
  { x: 2009, y: 282, y1: 242, y2: 898, y3: 222 }, { x: 2010, y: 321, y1: 121, y2: 567, y3: 231 },
  { x: 2011, y: 333, y1: 343, y2: 456, y3: 122 }, { x: 2012, y: 351, y1: 451, y2: 345, y3: 333 },
  { x: 2013, y: 403, y1: 203, y2: 543, y3: 354 }, { x: 2014, y: 421, y1: 431, y2: 654, y3: 100 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    title: 'Year',
    minimum: 2006,
    maximum: 2015,
    interval: 1
  },
  primaryYAxis: {
    title: 'Sales Percentage(%)'
  },
  series: [
    {
      type: 'StackingBar', name: 'John',
      //Stacking group for stacked bar
      stackingGroup: 'JohnAndAndrew',
      dataSource: chartData, xName: 'x', yName: 'y'
    }, {
      type: 'StackingBar', name: 'Andrew', stackingGroup: 'JohnAndAndrew',
      dataSource: chartData, xName: 'x', yName: 'y1'
    }, {
      type: 'StackingBar', name: 'Thomas', stackingGroup: 'ThomasAndMichael',
      dataSource: chartData, xName: 'x', yName: 'y2'
    }, {
      type: 'StackingBar', name: 'Michael', stackingGroup: 'ThomasAndMichael',
      dataSource: chartData, xName: 'x', yName: 'y3'
    }
  ],
  title: 'Sales by year'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Cylindrical stacked bar chart

To render a cylindrical stacked bar chart, set the [columnFacet](#) property to **Cylinder** in the chart series.

INDEX.JS

```

var cylindricalData = [
    { x: 2000, y: 0.61, y1: 0.03, y2: 0.48 }, { x: 2001, y: 0.81, y1: 0.05,
y2: 0.53 },
    { x: 2002, y: 0.91, y1: 0.06, y2: 0.57 }, { x: 2003, y: 1, y1: 0.09, y2:
0.61 },
    { x: 2004, y: 1.19, y1: 0.14, y2: 0.63 }, { x: 2005, y: 1.47, y1: 0.20,
y2: 0.64 },
    { x: 2006, y: 1.74, y1: 0.29, y2: 0.66 }, { x: 2007, y: 1.98, y1: 0.46,
y2: 0.76 },
    { x: 2008, y: 1.99, y1: 0.64, y2: 0.77 }, { x: 2009, y: 1.70, y1: 0.75,
y2: 0.55 }
];
var chart = new ej.charts.Chart({
    series: [
        {
            //Series type as stacked bar with cylinder shape
            type: 'StackingBar', columnFacet: 'Cylinder',
            dataSource: cylindricalData, xName: 'x', yName: 'y'
        },
        {
            type: 'StackingBar', columnFacet: 'Cylinder',
            dataSource: cylindricalData, xName: 'x', yName: 'y1'
        },
        {
            type: 'StackingBar', columnFacet: 'Cylinder',
            dataSource: cylindricalData, xName: 'x', yName: 'y2'
        }
    ],
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Series customization

The following properties can be used to customize the **stacked bar** series.

- [fill](#) – Specifies the color of the series.
- [opacity](#) – Specifies the opacity of [fill](#).
- [dashArray](#) – Specifies the dashes of series.
- [border](#) – Specifies the [color](#) and [width](#) of series border.

INDEX.JS

```

var chartData = [
  { x: 'Jan', y: 6, y1: 6, y2: -1 },
  { x: 'Feb', y: 8, y1: 8, y2: -1.5 },
  { x: 'Mar', y: 12, y1: 11, y2: -2 },
  { x: 'Apr', y: 15, y1: 16, y2: -2.5 },
  { x: 'May', y: 20, y1: 21, y2: -3 },
  { x: 'Jun', y: 24, y1: 25, y2: -3.5 },
  { x: 'Jul', y: 28, y1: 27, y2: -4 },
  { x: 'Aug', y: 32, y1: 31, y2: -4.5 },
  { x: 'Sep', y: 33, y1: 34, y2: -5 },
  { x: 'Oct', y: 35, y1: 34, y2: -5.5 },
  { x: 'Nov', y: 40, y1: 41, y2: -6 },
  { x: 'Dec', y: 42, y1: 42, y2: -6.5 },
];
var chart = new ej.charts.Chart(

```

```

{
  primaryXAxis: {
    valueType: 'Category',
    title: 'Months',
  },
  primaryYAxis: {
    title: 'Percentage (%)',
    minimum: -20,
    maximum: 100,
    labelFormat: '{value}%',
    edgeLabelPlacement: 'Shift',
  },
  series: [
    {
      //Series type as stacked bar
      type: 'StackingBar',
      name: 'Apple',
      dataSource: chartData,
      xName: 'x',
      yName: 'y',
      border: { width: 2, color: 'red' },
    },
    {
      type: 'StackingBar',
      name: 'Orange',
      dataSource: chartData,
      xName: 'x',
      yName: 'y1',
      border: { width: 2, color: 'grey' },
    },
    {
      type: 'StackingBar',
      name: 'Wastage',
      dataSource: chartData,
      xName: 'x',
      yName: 'y2',
      border: { width: 2, color: 'lime' },
    },
  ],
  title: 'Sales Comparison',
},
'#element'
);

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See also

- [Data label](#)
- [Tooltip](#)

Stacked bar Chart in ##Platform_Name## control

100% Stacked bar

INDEX.JS

```

var chartData = [
    { x: 'Jan', y: 6, y1: 6, y2: -1 }, { x: 'Feb', y: 8, y1: 8, y2: -1.5 },
    { x: 'Mar', y: 12, y1: 11, y2: -2 }, { x: 'Apr', y: 15, y1: 16, y2: -2.5 },
    { x: 'May', y: 20, y1: 21, y2: -3 }, { x: 'Jun', y: 24, y1: 25, y2: -3.5 },
    { x: 'Jul', y: 28, y1: 27, y2: -4 }, { x: 'Aug', y: 32, y1: 31, y2: -4.5 },
    { x: 'Sep', y: 33, y1: 34, y2: -5 }, { x: 'Oct', y: 35, y1: 34, y2: -5.5 },
    { x: 'Nov', y: 40, y1: 41, y2: -6 }, { x: 'Dec', y: 42, y1: 42, y2: -6.5 },
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        valueType: 'Category',
        title: 'Months'
    },
    primaryYAxis: {
        title: 'Percentage (%)',
        minimum: -20,
        maximum: 100,
        labelFormat: '{value}%',
        edgeLabelPlacement: 'Shift'
    },

```

```

series: [
    {
        //Series type as 100% stacked bar
        type: 'StackingBar100',
        name: 'Apple',
        dataSource: chartData, xName: 'x', yName: 'y'
    }, {
        type: 'StackingBar100', name: 'Orange',
        dataSource: chartData, xName: 'x', yName: 'y1'
    }, {
        type: 'StackingBar100', name: 'Wastage',
        dataSource: chartData, xName: 'x', yName: 'y2'
    }
],
title: 'Sales Comparison'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

100% Cylindrical stacked bar chart

To render a 100% cylindrical stacked bar chart, set the [columnFacet](#) property to **Cylinder** in the chart series.

INDEX.JS

```

var cylindricalData = [

```

```

    { x: 2000, y: 0.61, y1: 0.03, y2: 0.48 }, { x: 2001, y: 0.81, y1: 0.05,
y2: 0.53 },
    { x: 2002, y: 0.91, y1: 0.06, y2: 0.57 }, { x: 2003, y: 1, y1: 0.09, y2:
0.61 },
    { x: 2004, y: 1.19, y1: 0.14, y2: 0.63 }, { x: 2005, y: 1.47, y1: 0.20,
y2: 0.64 },
    { x: 2006, y: 1.74, y1: 0.29, y2: 0.66 }, { x: 2007, y: 1.98, y1: 0.46,
y2: 0.76 },
    { x: 2008, y: 1.99, y1: 0.64, y2: 0.77 }, { x: 2009, y: 1.70, y1: 0.75,
y2: 0.55 }
];
var chart = new ej.charts.Chart({
    series: [
        {
            //Series type as 100% stacked bar with cylindrical shape
            type: 'StackingBar100', columnFacet: 'Cylinder',
            dataSource: cylindricalData, xName: 'x', yName: 'y'
        },
        {
            type: 'StackingBar100', columnFacet: 'Cylinder',
            dataSource: cylindricalData, xName: 'x', yName: 'y1'
        },
        {
            type: 'StackingBar100', columnFacet: 'Cylinder',
            dataSource: cylindricalData, xName: 'x', yName: 'y2'
        }
    ],
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>

```

```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Series customization

The following properties can be used to customize the 100% stacked column series.

- [fill](#) – Specifies the color of the series.
- [opacity](#) – Specifies the opacity of [fill](#).
- [dashArray](#) – Specifies the dashes of series.
- [border](#) – Specifies the [color](#) and [width](#) of series border.

INDEX.JS

```
var chartData = [
  { x: 'Jan', y: 6, y1: 6, y2: 1 },
  { x: 'Feb', y: 8, y1: 8, y2: 1.5 },
  { x: 'Mar', y: 12, y1: 11, y2: 2 },
  { x: 'Apr', y: 15, y1: 16, y2: 2.5 },
  { x: 'May', y: 20, y1: 21, y2: 3 },
  { x: 'Jun', y: 24, y1: 25, y2: 3.5 },
  { x: 'Jul', y: 28, y1: 27, y2: 4 },
  { x: 'Aug', y: 32, y1: 31, y2: 4.5 },
  { x: 'Sep', y: 33, y1: 34, y2: 5 },
  { x: 'Oct', y: 35, y1: 34, y2: 5.5 },
  { x: 'Nov', y: 40, y1: 41, y2: 6 },
  { x: 'Dec', y: 42, y1: 42, y2: 6.5 },
];
var chart = new ej.charts.Chart(
{
  primaryXAxis: {
    valueType: 'Category',
    title: 'Months',
  },
  primaryYAxis: {
    title: 'Percentage (%)',

    labelFormat: '{value}%',
    edgeLabelPlacement: 'Shift',
  },
  series: [
    {
      //Series type as 100% stacked bar
      type: 'StackingBar100',
      name: 'Apple',
      dataSource: chartData,
      xName: 'x',
      yName: 'y', border: { width: 1.5, color: 'red' },
    },
    {
      type: 'StackingBar100',
      name: 'Orange',
      dataSource: chartData,
      xName: 'x',
      yName: 'y1', border: { width: 1.5, color: 'red' },
    }
  ]
}
```



```

    },
    {
      type: 'StackingBar100',
      name: 'Wastage',
      dataSource: chartData,
      xName: 'x',
      yName: 'y2', border: { width: 1.5, color: 'red' },
    },
  ],
  title: 'Sales Comparison',
},
'#element'
);

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

See also

- [Data label](#)
- [Tooltip](#)

Scatter Chart in ##Platform_Name## control

Scatter Charts

To render a scatter series, use series [type](#) as `Scatter` and inject `ScatterSeries` module using `Chart.Inject(ScatterSeries)` method.

INDEX.JS

```

var series1 = [];
var series2 = [];
var point1;
var value = 80;
var value1 = 70;
var i;
for (i = 1; i < 120; i++) {
    if (Math.random() > 0.5) {
        value += Math.random();
    } else {
        value -= Math.random();
    }
    value = value < 60 ? 60 : value > 90 ? 90 : value;
    point1 = { x: 120 + (i / 2), y: value.toFixed(1) };
    series1.push(point1);
}
for (i = 1; i < 120; i++) {
    if (Math.random() > 0.5) {
        value1 += Math.random();
    } else {
        value1 -= Math.random();
    }
    value1 = value1 < 60 ? 60 : value1 > 90 ? 90 : value1;
    point1 = { x: 120 + (i / 2), y: value1.toFixed(1) };
    series2.push(point1);
}
var chart = new ej.charts.Chart({
    primaryXAxis: {
        title: 'Height (cm)',
        minimum: 120, maximum: 180,
        edgeLabelPlacement: 'Shift',
        labelFormat: '{value}cm'
    },
    primaryYAxis: {
        title: 'Weight (kg)',
        minimum: 60, maximum: 90,
        labelFormat: '{value}kg',
        rangePadding: 'None'
    },
    series: [
        {
            //Series type as scatter
            type: 'Scatter',
            dataSource: series1, xName: 'x', yName: 'y',
            name: 'Male', opacity : 0.7,
            marker: { width: 10, height: 10 }
        }, {
            type: 'Scatter',
            dataSource: series2, xName: 'x', yName: 'y',
            name: 'Female', opacity : 0.7,
            marker: { width: 10, height: 10 }
        }
    ],
    title: 'Height Vs Weight'
});

```

```
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Series customization

The following properties can be used to customize the **scatter** series.

- [fill](#) – Specifies the color of the series.
- [opacity](#) – Specifies the opacity of [fill](#).
- [shape](#) - Specifies the shape of the scatter series.

INDEX.JS

```
var series1 = [];
var series2 = [];
var point1;
var value = 80;
var value1 = 70;
var i;
for (i = 1; i < 120; i++) {
  if (Math.random() > 0.5) {
    value += Math.random();
  } else {
    value -= Math.random();
  }
  value = value < 60 ? 60 : value > 90 ? 90 : value;
```

```

point1 = { x: 120 + i / 2, y: value.toFixed(1) };
series1.push(point1);
}
for (i = 1; i < 120; i++) {
  if (Math.random() > 0.5) {
    value1 += Math.random();
  } else {
    value1 -= Math.random();
  }
  value1 = value1 < 60 ? 60 : value1 > 90 ? 90 : value1;
  point1 = { x: 120 + i / 2, y: value1.toFixed(1) };
  series2.push(point1);
}
var chart = new ej.charts.Chart(
{
  primaryXAxis: {
    title: 'Height (cm)',
    minimum: 120,
    maximum: 180,
    edgeLabelPlacement: 'Shift',
    labelFormat: '{value}cm',
  },
  primaryYAxis: {
    title: 'Weight (kg)',
    minimum: 60,
    maximum: 90,
    labelFormat: '{value}kg',
    rangePadding: 'None',
  },
  series: [
    {
      //Series type as scatter
      type: 'Scatter',
      dataSource: series1,
      xName: 'x',
      yName: 'y',
      name: 'Male',
      opacity: 0.7,
      marker: { width: 5, height: 5, shape: 'Triangle' },
    },
    {
      type: 'Scatter',
      dataSource: series2,
      xName: 'x',
      yName: 'y',
      name: 'Female',
      opacity: 0.7,
      fill: 'red',
      marker: { width: 3, height: 3, shape: 'Square' },
    },
  ],
  title: 'Height Vs Weight',
},
'#element'
);

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Data label](#)
- [Tooltip](#)

Bubble Chart in ##Platform_Name## control

Bubble

INDEX.JS

```

var chart = new ej.charts.Chart({
  //Initializing Primary X Axis
  primaryXAxis: {
    title: 'Literacy Rate',
    minimum: 60,
    maximum: 100,
    interval: 5
  },
  //Initializing Primary Y Axis
  primaryYAxis: {
    title: 'GDP growth rate',
    minimum: -2,
    maximum: 16,
    interval: 2
  },
  //Initializing Chart Series

```

```

series: [
  {
    type: 'Bubble',
    dataSource: [{ x: 92.2, y: 7.8, size: 1.347, text: 'China' },
    { x: 74, y: 6.5, size: 1.241, text: 'India' },
    { x: 90.4, y: 6.0, size: 0.238, text: 'Indonesia' },
    { x: 99.4, y: 2.2, size: 0.312, text: 'US' },
    { x: 88.6, y: 1.3, size: 0.197, text: 'Brazil' },
    { x: 99, y: 0.7, size: 0.0818, text: 'Germany' },
    { x: 72, y: 2.0, size: 0.0826, text: 'Egypt' },
    { x: 99.6, y: 3.4, size: 0.143, text: 'Russia' },
    { x: 99, y: 0.2, size: 0.128, text: 'Japan' },
    { x: 86.1, y: 4.0, size: 0.115, text: 'Mexico' },
    { x: 92.6, y: 6.6, size: 0.096, text: 'Philippines' },
    { x: 61.3, y: 14.5, size: 0.162, text: 'Nigeria' }],
    xName: 'x', yName: 'y', size: 'size', name: 'pound',
  },
],
title: 'GDP vs Literacy Rate',
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Bubble Size Mapping

`size` property can be used to map the size value specified in data source.

INDEX.JS

```

var chart = new ej.charts.Chart({
  //Initializing Primary X Axis
  primaryXAxis: {
    title: 'Literacy Rate',
    minimum: 60,
    maximum: 100,
    interval: 5
  },
  //Initializing Primary Y Axis
  primaryYAxis: {
    title: 'GDP growth rate',
    minimum: -2,
    maximum: 16,
    interval: 2
  },
  //Initializing Chart Series
  series: [
    {
      type: 'Bubble',
      dataSource: [{ x: 92.2, y: 7.8, size: 1.347, text: 'China' },
        { x: 74, y: 6.5, size: 1.241, text: 'India' },
        { x: 90.4, y: 6.0, size: 0.238, text: 'Indonesia' },
        { x: 99.4, y: 2.2, size: 0.312, text: 'US' },
        { x: 88.6, y: 1.3, size: 0.197, text: 'Brazil' },
        { x: 99, y: 0.7, size: 0.0818, text: 'Germany' },
        { x: 72, y: 2.0, size: 0.0826, text: 'Egypt' },
        { x: 99.6, y: 3.4, size: 0.143, text: 'Russia' },
        { x: 99, y: 0.2, size: 0.128, text: 'Japan' },
        { x: 86.1, y: 4.0, size: 0.115, text: 'Mexico' },
        { x: 92.6, y: 6.6, size: 0.096, text: 'Philippines' },
        { x: 61.3, y: 14.5, size: 0.162, text: 'Nigeria' }],
      xName: 'x', yName: 'y', size: 'size', name: 'pound',
    },
  ],
  title: 'GDP vs Literacy Rate',
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>

```

```
<body>

  <div id="container">
    <div id="element"></div>
  </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Series customization

The following properties can be used to customize the **bubble** series.

- [fill](#) – Specifies the color of the series.
- [opacity](#) – Specifies the opacity of [fill](#).

INDEX.JS

```
var chart = new ej.charts.Chart({
  //Initializing Primary X Axis
  primaryXAxis: {
    title: 'Literacy Rate',
    minimum: 60,
    maximum: 100,
    interval: 5
  },
  //Initializing Primary Y Axis
  primaryYAxis: {
    title: 'GDP growth rate',
    minimum: -2,
    maximum: 16,
    interval: 2
  },
  //Initializing Chart Series
  series: [
    {
      type: 'Bubble',
      dataSource: [{ x: 92.2, y: 7.8, size: 1.347, text: 'China' },
        { x: 74, y: 6.5, size: 1.241, text: 'India' },
        { x: 90.4, y: 6.0, size: 0.238, text: 'Indonesia' },
        { x: 99.4, y: 2.2, size: 0.312, text: 'US' },
        { x: 88.6, y: 1.3, size: 0.197, text: 'Brazil' },
        { x: 99, y: 0.7, size: 0.0818, text: 'Germany' },
        { x: 72, y: 2.0, size: 0.0826, text: 'Egypt' },
        { x: 99.6, y: 3.4, size: 0.143, text: 'Russia' },
        { x: 99, y: 0.2, size: 0.128, text: 'Japan' },
        { x: 86.1, y: 4.0, size: 0.115, text: 'Mexico' },
        { x: 92.6, y: 6.6, size: 0.096, text: 'Philippines' },
        { x: 61.3, y: 14.5, size: 0.162, text: 'Nigeria' }],
      xName: 'x', yName: 'y', size: 'size', name: 'pound',
    }
  ],
});
```



```

        fill: 'blue', opacity: 0.5
    },
    ],
    title: 'GDP vs Literacy Rate',
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Data label](#)
- [Tooltip](#)

Polar Chart in ##Platform_Name## control

Polar Chart

To render a polar series, use series [type](#) as **Polar** and inject **PolarSeries** module using **Chart.Inject(PolarSeries)** method.

Draw Types

Polar drawType property is used to change the series plotting type to line, column, area, range column, spline, scatter, stacking area and stacking column. The default value of drawType is **Line**.

Line

To render a line draw type, use series [drawType](#) as **Line** and inject **LineSeries** module using **Chart.Inject(LineSeries)** method.

[isClosed](#) property specifies whether to join start and end point of a line series used in polar chart to form a closed path. Default value of isClosed is true.

INDEX.JS

```
var chartData = [
    { x: 2005, y: 28 }, { x: 2006, y: 25 }, { x: 2007, y: 26 }, { x: 2008, y:
27 },
    { x: 2009, y: 32 }, { x: 2010, y: 35 }, { x: 2011, y: 30 }
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        title: 'Year',
        minimum: 2004, maximum: 2012, interval: 1
    },
    primaryYAxis: {
        minimum: 20, maximum: 40, interval: 5,
        title: 'Efficiency',
        labelFormat: '{value}%'
    },
    series:[{
        dataSource: chartData, width:2,
        xName: 'x', yName: 'y',
        name: 'India',
        //Series type as polar
        type: 'Polar',
        // Series draw type as line
        drawType: 'Line'
    }],
    title: 'Efficiency of oil-fired power production'
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
```

```

    ele.style.visibility = "visible";
  }
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Spline

To render a spline draw type, use series [drawType](#) as `Spline` and inject `SplineSeries` module using `Chart.Inject(SplineSeries)` method.

INDEX.JS

```

var chartData = [
  { x: 'Jan', y: -1 }, { x: 'Feb', y: -1 }, { x: 'Mar', y: 2 },
  { x: 'Apr', y: 8 }, { x: 'May', y: 13 }, { x: 'Jun', y: 18 },
  { x: 'Jul', y: 21 }, { x: 'Aug', y: 20 }, { x: 'Sep', y: 16 },
  { x: 'Oct', y: 10 }, { x: 'Nov', y: 4 }, { x: 'Dec', y: 0 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    title: 'Month',
    valueType: 'Category'
  },
  primaryYAxis: {
    minimum: -5, maximum: 35, interval: 10,
    title: 'Temperature in Celsius',
    labelFormat: '{value}C'
  },
  series:[{
    dataSource: chartData, width:2,
    xName: 'x', yName: 'y',
    name: 'London',
    // Series type as Polar series
    type: 'Polar',
    // Series draw type as spline
    drawType: 'Spline'
  }],
  title: 'Climate Graph-2012'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>

```

```

</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Area

To render a area draw type, use series [drawType](#) as `Area` and inject `AreaSeries` module using `Chart.Inject(AreaSeries)` method.

INDEX.JS

```

var chartData = [
    { x: 1900, y: 4 }, { x: 1920, y: 3.0 }, { x: 1940, y: 3.8 },
    { x: 1960, y: 3.4 }, { x: 1980, y: 3.2 }, { x: 2000, y: 3.9 }
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        title: 'Year',
        minimum: 1900, maximum: 2000, interval: 10,
        edgeLabelPlacement: 'Shift'
    },
    primaryYAxis: {
        minimum: 2, maximum: 5, interval: 0.5,
        title: 'Sales Amount in Millions'
    },
    series:[{
        dataSource: chartData,
        xName: 'x', yName: 'y',
        opacity: 0.5, fill: '#69D2E7',
        name: 'Product A',
        // Series type as polar series
        type: 'Polar',
        // Series draw type as area
        drawType: 'Area'
    }],
    title: 'Average Sales Comparison'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">

```

```

<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Stacked Area

To render a stacked area draw type, use series [drawType](#) as `StackingArea` and inject `StackingAreaSeries` module using `Chart.Inject(StackingAreaSeries)` method.

INDEX.JS

```

var chartData = [
    { x: '2000', y: 0.61, y1: 0.03, y2: 0.48 },
    { x: '2001', y: 0.81, y1: 0.05, y2: 0.53 },
    { x: '2002', y: 0.91, y1: 0.06, y2: 0.57 },
    { x: '2003', y: 1, y1: 0.09, y2: 0.61 },
    { x: '2004', y: 1.19, y1: 0.14, y2: 0.63 },
    { x: '2005', y: 1.47, y1: 0.20, y2: 0.64 },
    { x: '2006', y: 1.74, y1: 0.29, y2: 0.66 },
    { x: '2007', y: 1.98, y1: 0.46, y2: 0.76 },
    { x: '2008', y: 1.99, y1: 0.64, y2: 0.77 },
    { x: '2009', y: 1.70, y1: 0.75, y2: 0.55 },
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        title: 'Years',
        valueType: 'Category',
        majorTickLines: { width: 0 },
        edgeLabelPlacement: 'Shift'
    },
    primaryYAxis: {
        title: 'Spend in Billions',
        minimum: 0,
        maximum: 4,
        interval: 1,
        majorTickLines: { width: 0 },
        labelFormat: '{value}B'
    }
});

```

```

    },
    series: [
        {
            dataSource: chartData, xName: 'x', yName: 'y',
            // Series type as polar series
            type : 'Polar',
            //Series draw type as stacked area series
            drawType: 'StackingArea',
            name: 'Organic',
        }, {
            dataSource: chartData, xName: 'x', yName: 'y1',
            type : 'Polar',
            drawType: 'StackingArea', name: 'Fair-trade',
        }, {
            dataSource: chartData, xName: 'x', yName: 'y2',
            type : 'Polar',
            drawType: 'StackingArea', name: 'Veg Alternatives',
        },
    ],
    title: 'Trend in Sales of Ethical Produce'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

Column

To render a column draw type, use series [drawType](#) as `Column`.

INDEX.JS

```

var chartData = [
    { country: "USA", gold: 50 },
    { country: "China", gold: 40 },
    { country: "Japan", gold: 70 },
    { country: "Australia", gold: 60 },
    { country: "France", gold: 50 },
    { country: "Germany", gold: 40 },
    { country: "Italy", gold: 40 },
    { country: "Sweden", gold: 30 }
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        valueType: 'Category',
        title: 'Countries'
    },
    primaryYAxis: {
        minimum: 0, maximum: 80,
        interval: 20, title: 'Medals'
    },
    series:[{
        dataSource: chartData,
        xName: 'country', yName: 'gold',
        name: 'Gold',
        // Series type as polar series
        type: 'Polar',
        // Series draw type as column series
        drawType: 'Column'
    }],
    title: 'Olympic Medals'
}, '#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
```

```

if(ele) {
    ele.style.visibility = "visible";
}

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Stacked Column

To render a stacked column draw type, use series [drawType](#) as `StackingColumn`.

INDEX.JS

```

var chartData = [
    { x: '2014', y: 111.1, y1: 76.9, y2: 66.1, y3: 34.1 },
    { x: '2015', y: 127.3, y1: 99.5, y2: 79.3, y3: 38.2 },
    { x: '2016', y: 143.4, y1: 121.7, y2: 91.3, y3: 44.0 },
    { x: '2017', y: 159.9, y1: 142.5, y2: 102.4, y3: 51.6 },
    { x: '2018', y: 175.4, y1: 166.7, y2: 112.9, y3: 61.9 },
    { x: '2019', y: 189.0, y1: 182.9, y2: 122.4, y3: 71.5 },
    { x: '2020', y: 202.7, y1: 197.3, y2: 120.9, y3: 82.0 }
];

var chart = new ej.charts.Chart({
    primaryXAxis: {
        title: 'Years',
        interval: 1,
        valueType: 'Category'
    },
    primaryYAxis: {
        title: 'Sales in Billions',
        minimum: 0,
        maximum: 700,
        interval: 100,
        labelFormat: '{value}B',
    },
    series: [
        {
            dataSource: chartData, xName: 'x', yName: 'y',
            type: 'Polar',
            //Series draw type as stacked column
            drawType: 'StackingColumn',
            name: 'UK',
        }, {
            dataSource: chartData, xName: 'x', yName: 'y1',
            type: 'Polar',
            drawType: 'StackingColumn', name: 'Germany',
        }, {
            dataSource: chartData, xName: 'x', yName: 'y2',
            type: 'Polar',
            drawType: 'StackingColumn', name: 'France',
        }, {
            dataSource: chartData, xName: 'x', yName: 'y3',
            type: 'Polar',
            drawType: 'StackingColumn', name: 'Italy',
        }
    ],
});

```



```

        title: 'Mobile Game Market by Country'
    }, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Range Column

To render a range column draw type, use series [drawType](#) as `RangeColumn`.

INDEX.JS

```

var data = [
  { x: 'Jan', low: 0.7, high: 6.1 }, { x: 'Feb', low: 1.3, high: 6.3 }, {
x: 'Mar', low: 1.9, high: 8.5 },
  { x: 'Apr', low: 3.1, high: 10.8 }, { x: 'May', low: 5.7, high: 14.40 },
{ x: 'Jun', low: 8.4, high: 16.90 },
  { x: 'Jul', low: 10.6, high: 19.20 }, { x: 'Aug', low: 10.5, high: 18.9 },
{ x: 'Sep', low: 8.5, high: 16.1 },
  { x: 'Oct', low: 6.0, high: 12.5 }, { x: 'Nov', low: 1.5, high: 6.9 },
{ x: 'Dec', low: 5.1, high: 12.1 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    title: 'Months',
    valueType: 'Category'
  },
  primaryYAxis: {
    title: 'Temperature(Celsius)',

```

```

    },
    series: [
        {
            type: 'Polar',
            // Series draw type as range column series
            drawType: 'RangeColumn',
            name: 'India',
            dataSource: data, xName: 'x', high: 'high', low: 'low',
        }
    ],
    title: 'Maximum and Minimum Temperature'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Scatter

To render a scatter draw type, use series [DrawType](#) as `Scatter` and inject `ScatterSeries` module using `Chart.Inject(ScatterSeries)` method.

INDEX.JS

```

var chartData = [
  { x: 'Jan', y: -1 }, { x: 'Feb', y: -1 }, { x: 'Mar', y: 2 },
  { x: 'Apr', y: 8 }, { x: 'May', y: 13 }, { x: 'Jun', y: 18 },
  { x: 'Jul', y: 21 }, { x: 'Aug', y: 20 }, { x: 'Sep', y: 16 },
  { x: 'Oct', y: 10 }, { x: 'Nov', y: 4 }, { x: 'Dec', y: 0 }
];

```

```

var chart = new ej.charts.Chart({
  primaryXAxis: {
    title: 'Month',
    valueType: 'Category'
  },
  primaryYAxis: {
    minimum: -5, maximum: 35, interval: 10,
    title: 'Temperature in Celsius',
    labelFormat: '{value}C'
  },
  series: [{
    dataSource: chartData, width: 2,
    xName: 'x', yName: 'y',
    name: 'London',
    // Series type as Polar series
    type: 'Polar',
    // Series draw type as scatter
    drawType: 'Scatter'
  }],
  title: 'Climate Graph-2012'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

*Series customization**Start Angle*

You can customize the start angle of the polar series using [startAngle](#) property. By default, `startAngle` is 0 degree.

INDEX.JS

```
var chartData = [
    { country: "USA", gold: 50 },
    { country: "China", gold: 40 },
    { country: "Japan", gold: 70 },
    { country: "Australia", gold: 60 },
    { country: "France", gold: 50 },
    { country: "Germany", gold: 40 },
    { country: "Italy", gold: 40 },
    { country: "Sweden", gold: 30 }
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        valueType: 'Category',
        title: 'Countries',
        startAngle: 90,
    },
    primaryYAxis: {
        minimum: 0, maximum: 80,
        interval: 20, title: 'Medals'
    },
    series:[{
        dataSource: chartData,
        xName: 'country', yName: 'gold',
        name: 'Gold',
        // Series type as polar series
        type: 'Polar',
        // Series draw type as column series
        drawType: 'Column'
    }],
    title: 'Olympic Medals'
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>
```

```

    <div id="container">
      <div id="element"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Radius

You can customize the radius of the polar series using [coefficient](#) property. By default, `coefficient` is 100.

INDEX.JS

```

var chartData = [
  { country: "USA", gold: 50 },
  { country: "China", gold: 40 },
  { country: "Japan", gold: 70 },
  { country: "Australia", gold: 60 },
  { country: "France", gold: 50 },
  { country: "Germany", gold: 40 },
  { country: "Italy", gold: 40 },
  { country: "Sweden", gold: 30 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    valueType: 'Category',
    title: 'Countries',
    coefficient: 80
  },
  primaryYAxis: {
    minimum: 0, maximum: 80,
    interval: 20, title: 'Medals'
  },
  series:[{
    dataSource: chartData,
    xName: 'country', yName: 'gold',
    name: 'Gold',
    // Series type as polar series
    type: 'Polar',
    // Series draw type as column series
    drawType: 'Column'
  }],
  title: 'Olympic Medals'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>

```

```

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Data label](#)
- [Tooltip](#)

Radar Chart in ##Platform_Name## control

Radar Chart

To render a radar series, use series [type](#) as **Radar** and inject **PolarSeries** module using **Chart.Inject(RadarSeries)** method.

Draw Type

similar to Polar drawType, Radar draw property is used to change the series plotting type to line, column, area, range column, spline, scatter, stacking area and stacking column. The default value of drawType is **Line**.

INDEX.JS

```

var chartData = [
    { x: 2005, y: 28 }, { x: 2006, y: 25 }, { x: 2007, y: 26 }, { x: 2008, y:
27 },
    { x: 2009, y: 32 }, { x: 2010, y: 35 }, { x: 2011, y: 30 }
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        title: 'Year',
        minimum: 2004, maximum: 2012, interval: 1
    },
    primaryYAxis: {

```

```

        minimum: 20, maximum: 40, interval: 5,
        title: 'Efficiency',
        labelFormat: '{value}%'
    },
    series:[{
        dataSource: chartData, width:2,
        xName: 'x', yName: 'y',
        name: 'India',
        //Series type as polar
        type: 'Radar',
        // Series draw type as line
        drawType: 'Line'
    }],
    title: 'Efficiency of oil-fired power production'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Customization

Start Angle

You can customize the start angle of the polar series using [startAngle](#) property. By default, `startAngle` is 0 degree.

INDEX.JS

```

var data = [
  { country: "USA", gold: 50 },

```

```

    { country: "China", gold: 40 },
    { country: "Japan", gold: 70 },
    { country: "Australia", gold: 60 },
    { country: "France", gold: 50 },
    { country: "Germany", gold: 40 },
    { country: "Italy", gold: 40 },
    { country: "Sweden", gold: 30 }
  ];
  var chart = new ej.charts.Chart({
    primaryXAxis: {
      valueType: 'Category',
      title: 'Countries',
      startAngle: 90,
    },
    primaryYAxis: {
      minimum: 0, maximum: 80,
      interval: 20, title: 'Medals'
    },
    series: [{
      dataSource: data,
      xName: 'country', yName: 'gold',
      name: 'Gold',
      // Series type as radar series
      type: 'Radar',
      // Series draw type as column series
      drawType: 'Column'
    }],
    title: 'Olympic Medals'
  }, '#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
  var ele = document.getElementById('container');
  if(ele) {
    ele.style.visibility = "visible";
  }
  </script>
```



```

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Radius

You can customize the radius of the polar series using [coefficient](#) property. By default, `coefficient` is 100.

INDEX.JS

```

var data = [
    { country: "USA", gold: 50 },
    { country: "China", gold: 40 },
    { country: "Japan", gold: 70 },
    { country: "Australia", gold: 60 },
    { country: "France", gold: 50 },
    { country: "Germany", gold: 40 },
    { country: "Italy", gold: 40 },
    { country: "Sweden", gold: 30 }
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        valueType: 'Category',
        title: 'Countries',
        coefficient: 80
    },
    primaryYAxis: {
        minimum: 0, maximum: 80,
        interval: 20, title: 'Medals'
    },
    series:[{
        dataSource: data,
        xName: 'country', yName: 'gold',
        name: 'Gold',
        // Series type as radar series
        type: 'Radar',
        // Series draw type as column series
        drawType: 'Column'
    }],
    title: 'Olympic Medals'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>

```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Data label](#)
- [Tooltip](#)

High low Chart in ##Platform_Name## control

Hilo

Hilo Series illustrates the price movements in stock using the high and low values. To render a Hilo series, use series [type](#) as **Hilo** and inject **HiloSeries** module using **Chart.Inject(HiloSeries)** method.

Hilo series requires 3 fields (x, high and low) to show the high and low price in the stock.

INDEX.JS

```

var chartData = [
    { x: 'Jan', low: 87, high: 200 }, { x: 'Feb', low: 45, high: 135 },
    { x: 'Mar', low: 19, high: 85 }, { x: 'Apr', low: 31, high: 108 },
    { x: 'May', low: 27, high: 80 }, { x: 'June', low: 84, high: 130 },
    { x: 'July', low: 77, high: 150 }, { x: 'Aug', low: 54, high: 125 },
    { x: 'Sep', low: 60, high: 155 }, { x: 'Oct', low: 60, high: 180 },
    { x: 'Nov', low: 88, high: 180 }, { x: 'Dec', low: 84, high: 230 }
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        valueType: 'Category',
        title: 'Months'
    },
    primaryYAxis: {
        labelFormat: '{value}mm',
        edgeLabelPlacement: 'Shift',
        title: 'Rainfall',
    },
    series:[
        {
            dataSource: chartData, width:2,
            xName: 'x', high: 'high', low: 'low',
            name: 'India',

```

```

        //Series type as Hilo
        type: 'Hilo'
    },
    ],
    title: 'Maximum and Minimum Rainfall'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Series customization

The following properties can be used to customize the **hilo** series.

- [fill](#) – Specifies the color of the series.
- [opacity](#) – Specifies the opacity of [fill](#).

INDEX.JS

```

var chartData = [
  { x: 'Jan', low: 87, high: 200 }, { x: 'Feb', low: 45, high: 135 },
  { x: 'Mar', low: 19, high: 85 }, { x: 'Apr', low: 31, high: 108 },
  { x: 'May', low: 27, high: 80 }, { x: 'June', low: 84, high: 130 },
  { x: 'July', low: 77, high: 150 }, { x: 'Aug', low: 54, high: 125 },
  { x: 'Sep', low: 60, high: 155 }, { x: 'Oct', low: 60, high: 180 },
  { x: 'Nov', low: 88, high: 180 }, { x: 'Dec', low: 84, high: 230 }
];
var chart = new ej.charts.Chart({

```

```

primaryXAxis: {
    valueType: 'Category',
    title: 'Months'
},
primaryYAxis:
{
    labelFormat: '{value}mm',
    edgeLabelPlacement: 'Shift',
    title: 'Rainfall',
},
series:[
    {
        dataSource: chartData, width:2,
        xName: 'x', high: 'high', low: 'low',
        name: 'India', fill: 'blue',
        //Series type as Hilo
        type: 'Hilo'
    }
],
title: 'Maximum and Minimum Rainfall'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Data label](#)

- [Tooltip](#)

High low open close Chart in `##Platform_Name##` control

High Low Open Close

HiloOpenClose series is used to represent the low, high, open and closing values over time. To render a HiloOpenClose series, use series [type](#) as `HiloOpenClose` and inject `HiloOpenCloseSeries` module using `Chart.Inject(HiloOpenCloseSeries)` method.

HiloOpenClose series requires 5 fields (x, high, low, open and close) to show the high, low, open and close price values in the stock.

INDEX.JS

```
var chartData = [
  { x: 'Jan', open: 120, high: 160, low: 100, close: 140 },
  { x: 'Feb', open: 150, high: 190, low: 130, close: 170 },
  { x: 'Mar', open: 130, high: 170, low: 110, close: 150 },
  { x: 'Apr', open: 160, high: 180, low: 120, close: 140 },
  { x: 'May', open: 150, high: 170, low: 110, close: 130 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    title: 'Date',
    valueType: 'Category',
  },
  primaryYAxis: {
    title: 'Price in Dollar', minimum: 100, maximum: 200, interval: 20,
  },
  series:[
    {
      dataSource: chartData, width:2,
      xName: 'x', open: 'open', close: 'close', high: 'high', low:
'low',
      name: 'SHIRPUR-G',
      // Series type as HiloOpenClose
      type: 'HiloOpenClose'
    }
  ],
  title: 'Financial Analysis'
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Series customization

In **hiloOpenClose** series, **bullFillColor** property is used to fill the segment when the open value is greater than the close value and **bearFillColor** property is used to fill the segment when the open value is less than the close value. By default, **bullFillColor** is set as **green** and **bearFillColor** is set as **red**.

INDEX.JS

```

var chartData = [
    { x: 'Jan', open: 120, high: 160, low: 100, close: 140 },
    { x: 'Feb', open: 150, high: 190, low: 130, close: 170 },
    { x: 'Mar', open: 130, high: 170, low: 110, close: 150 },
    { x: 'Apr', open: 160, high: 180, low: 120, close: 140 },
    { x: 'May', open: 150, high: 170, low: 110, close: 130 }
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        title: 'Date',
        valueType: 'Category',
    },
    primaryYAxis: {
        title: 'Price in Dollar', minimum: 100, maximum: 200, interval: 20,
    },
    series: [
        {
            type: 'HiloOpenClose',
            // sets the bullFill and bearFill color of hiloopenclose chart
            bearFillColor: 'red',
            bullFillColor: 'green',
            dataSource: chartData, width: 2,
            xName: 'x', open: 'open', close: 'close', high: 'high', low:
'low',
            name: 'SHIRPUR-G'
        }
    ],
    title: 'Financial Analysis'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Data label](#)
- [Tooltip](#)

Candle Chart in ##Platform_Name## control

Candle

Candle series is similar to Hilo Open Close series, are used to represent the low, high, open and closing price over time. To render a candle series, use series [type](#) as **Candle** and inject **CandleSeries** module using **Chart.Inject(CandleSeries)** method.

INDEX.JS

```

var chartData = [
  { x: 'Jan', open: 120, high: 160, low: 100, close: 140 },
  { x: 'Feb', open: 150, high: 190, low: 130, close: 170 },
  { x: 'Mar', open: 130, high: 170, low: 110, close: 150 },
  { x: 'Apr', open: 160, high: 180, low: 120, close: 140 },
  { x: 'May', open: 150, high: 170, low: 110, close: 130 }
];
var Chart = new ej.charts.Chart({
  primaryXAxis: {
    title: 'Date',
    valueType: 'Category',

```

```

        majorGridLines: { width: 0 },
    },
    primaryYAxis:
    {
        title: 'Price',
        minimum: 100,
        maximum: 200,
        interval: 20,
    },
    series:[
        {
            dataSource: chartData, width:2,
            xName: 'x', open: 'open', close: 'close', high: 'high', low:
'low',
            name: 'SHIRPUR-G',
            // Series type as candle series
            type: 'Candle'
        }
    ],
    title: 'Shirpur Gold Refinery Share Price'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Hollow Candles

Candle charts allow to visually compare the current price with previous price by coloring them.

Candles are filled/left as hollow based on the following criteria.

<!-- markdownlint-disable MD033 -->

States	Description
Filled	candle sticks are filled when the close value is lesser than the open value
Unfilled	candle sticks are unfilled when the close value is greater than the open value

The color of the candle will be defined by comparing with previous values. Bear color will be applied when the current closing value is greater than the previous closing value. Bull color will be applied when the current closing value is less than the previous closing value.

By default, bullFillColor is set as red and bearFillColor is set as green.

Solid Candles

[enableSolidCandles](#) is used to enable/disable the solid candles. By default is set to be false. The fill color of the candle will be defined by its opening and closing values.

[bearFillColor](#) will be applied when the opening value is less than the closing value.

[bullFillColor](#) will be applied when the opening value is greater than closing value.

INDEX.JS

```
var chartData = [
  { x: 'Jan', open: 120, high: 160, low: 100, close: 140 },
  { x: 'Feb', open: 150, high: 190, low: 130, close: 170 },
  { x: 'Mar', open: 130, high: 170, low: 110, close: 150 },
  { x: 'Apr', open: 160, high: 180, low: 120, close: 140 },
  { x: 'May', open: 150, high: 170, low: 110, close: 130 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    title: 'Date',
    valueType: 'Category',
    majorGridLines: { width: 0 },
  },
  primaryYAxis: {
    title: 'Price',
    minimum: 100,
    maximum: 200,
    interval: 20,
  },
  series:[
    {
      dataSource: chartData, width:2,
      xName: 'x', open: 'open', close: 'close', high: 'high', low:
'low',
      name: 'SHIRPUR-G',
      bearFillColor: '#e56590',
      bullFillColor: '#f8b883',
      // Series type as candle series
      type: 'Candle'
    }
  ]
});
```

```

    ],
    title: 'Shirpur Gold Refinery Share Price'
  }, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Data label](#)
- [Tooltip](#)

Box whisker Chart in ##Platform_Name## control

Box and whisker

To render a box and whisker chart, use series [type](#) as `BoxAndWhisker` and inject

`BoxAndWhiskerSeries` module using `Chart.Inject(BoxAndWhiskerSeries)` method. The field `y` requires n number of data or it should contains minimum of five values to plot a segment.

INDEX.JS

```

var chart = new ej.charts.Chart({
  //Initializing Primary X Axis
  primaryXAxis: {
    valueType: 'Category',
  },
  //Initializing Chart Series

```

```

series: [
  {
    type: 'BoxAndWhisker',
    dataSource: [
      { x: 'Development', y: [22, 22, 23, 25, 25, 25, 26, 27, 27,
28, 28, 29, 30, 32, 34, 32, 34, 36, 35, 38] },
      { x: 'Testing', y: [22, 33, 23, 25, 26, 28, 29, 30, 34, 33,
32, 31, 50] },
      { x: 'HR', y: [22, 24, 25, 30, 32, 34, 36, 38, 39, 41, 35,
36, 40, 56] },
      { x: 'Finance', y: [26, 27, 28, 30, 32, 34, 35, 37, 35, 37,
45] },
      { x: 'R&D', y: [26, 27, 29, 32, 34, 35, 36, 37, 38, 39, 41,
43, 58] },
      { x: 'Sales', y: [27, 26, 28, 29, 29, 29, 32, 35, 32, 38,
53] },
      { x: 'Inventory', y: [21, 23, 24, 25, 26, 27, 28, 30, 34,
36, 38] },
      { x: 'Graphics', y: [26, 28, 29, 30, 32, 33, 35, 36, 52] },
      { x: 'Training', y: [28, 29, 30, 31, 32, 34, 35, 36] }
    ],
    xName: 'x',
    yName: 'y',
    marker: {
      visible: true,
      width: 10,
      height: 10
    },
  },
],
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {

```

```

    ele.style.visibility = "visible";
  }
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Box Plot

You can change the rendering mode of the Box and Whisker series using the `boxPlotMode` property. The default `boxPlotMode` is `exclusive`. The other `boxPlotMode` available are `inclusive` and `normal`.

INDEX.JS

```

var chart = new ej.charts.Chart({
  //Initializing Primary X Axis
  primaryXAxis: {
    valueType: 'Category',
  },
  //Initializing Chart Series
  series: [
    {
      type: 'BoxAndWhisker',
      boxPlotMode: 'Normal',
      dataSource: [
        { x: 'Development', y: [22, 22, 23, 25, 25, 25, 26, 27, 27, 28, 28, 29, 30, 32, 34, 32, 34, 36, 35, 38] },
        { x: 'Testing', y: [22, 33, 23, 25, 26, 28, 29, 30, 34, 33, 32, 31, 50] },
        { x: 'HR', y: [22, 24, 25, 30, 32, 34, 36, 38, 39, 41, 35, 36, 40, 56] },
        { x: 'Finance', y: [26, 27, 28, 30, 32, 34, 35, 37, 35, 37, 45] },
        { x: 'R&D', y: [26, 27, 29, 32, 34, 35, 36, 37, 38, 39, 41, 43, 58] },
        { x: 'Sales', y: [27, 26, 28, 29, 29, 29, 32, 35, 32, 38, 53] },
        { x: 'Inventory', y: [21, 23, 24, 25, 26, 27, 28, 30, 34, 36, 38] },
        { x: 'Graphics', y: [26, 28, 29, 30, 32, 33, 35, 36, 52] },
        { x: 'Training', y: [28, 29, 30, 31, 32, 34, 35, 36] }
      ],
      xName: 'x',
      yName: 'y',
      marker: {
        visible: true,
        width: 10,
        height: 10
      },
    },
  ],
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>

```

```

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Show mean

In Box and Whisker series **showMean** property is used to show the box and whisker average value. The default value of **showMean** is false.

INDEX.JS

```

var chart = new ej.charts.Chart({
    //Initializing Primary X Axis
    primaryXAxis: {
        valueType: 'Category',
    },
    //Initializing Chart Series
    series: [
        {
            type: 'BoxAndWhisker',
            dataSource: [
                { x: 'Development', y: [22, 22, 23, 25, 25, 25, 26, 27, 27,
28, 28, 29, 30, 32, 34, 32, 34, 36, 35, 38] },
                { x: 'Testing', y: [22, 33, 23, 25, 26, 28, 29, 30, 34, 33,
32, 31, 50] },
                { x: 'HR', y: [22, 24, 25, 30, 32, 34, 36, 38, 39, 41, 35,
36, 40, 56] },
                { x: 'Finance', y: [26, 27, 28, 30, 32, 34, 35, 37, 35, 37,
45] },
                { x: 'R&D', y: [26, 27, 29, 32, 34, 35, 36, 37, 38, 39, 41,
43, 58] },
                { x: 'Sales', y: [27, 26, 28, 29, 29, 29, 32, 35, 32, 38,
53] },
                { x: 'Inventory', y: [21, 23, 24, 25, 26, 27, 28, 30, 34,
36, 38] },
            ]
        }
    ]
});

```

```

        { x: 'Graphics', y: [26, 28, 29, 30, 32, 33, 35, 36, 52] },
        { x: 'Training', y: [28, 29, 30, 31, 32, 34, 35, 36] }
    ],
    showMean: false,
    xName: 'x',
    yName: 'y',
  }
],
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Data label](#)
- [Tooltip](#)

Waterfall Chart in ##Platform_Name## control

Waterfall Chart

Waterfall chart helps to understand the cumulative effect of the sequentially introduced positive and negative values. To render a Waterfall series, use series [type](#) as [Waterfall](#) and inject [WaterfallSeries](#) module using [Chart.Inject\(WaterfallSeries\)](#) method. [intermediateSumIndexes](#) property of waterfall is used to represent the in between sum values and [sumIndexes](#) is used to represent the cumulative sum values.

INDEX.JS

```

var chartData = [
  { x: 'Income', y: 4711 }, { x: 'Sales', y: -1015 },
  { x: 'Development', y: -688 },
  { x: 'Revenue', y: 1030 }, { x: 'Balance' },
  { x: 'Administrative', y: -780 },
  { x: 'Expense', y: -361 }, { x: 'Tax', y: -695 },
  { x: 'Net Profit' }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    valueType: 'Category',
  },
  series: [
    {
      dataSource: chartData,
      xName: 'x', yName: 'y', intermediateSumIndexes: [4], sumIndexes:
[8],
      //Series type as Waterfall
      type: 'Waterfall',
    }
  ],
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Series customization

The negative changes of waterfall charts are represented by using `negativeFillColor` and the summary changes are represented by using `summaryFillColor` properties respectively. By default, the `negativeFillColor` is **green** and the `summaryFillColor` is **black**.

INDEX.JS

```
var chartData = [
  { x: 'Income', y: 4711 }, { x: 'Sales', y: -1015 },
  { x: 'Development', y: -688 },
  { x: 'Revenue', y: 1030 }, { x: 'Balance', y: 0 },
  { x: 'Administrative', y: -780 },
  { x: 'Expense', y: -361 }, { x: 'Tax', y: -695 },
  { x: 'Net Profit', y: 0 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    valueType: 'Category',
    majorGridLines: {width: 0},
  },
  primaryYAxis: {
    labelFormat: '${value}M',
    minimum: 0, maximum: 5500, interval: 500,
    majorGridLines: {width: 0},
    lineStyle: { width: 0},
    majorTickLines: { width: 0}
  },
  series:[
    {
      dataSource: chartData, width:2,
      xName: 'x', yName:'y', intermediateSumIndexes: [4], sumIndexes:
[8],
      name: 'USA',columnWidth: 0.6,
      //Series type as Waterfall
      type: 'Waterfall', animation: { enable: true },
      marker: {
        dataLabel: { visible: true, position: 'Outer' }
      }, summaryFillColor: 'red',
      negativeFillColor: 'green',
      connector: { color: 'blue', width: 1.5 },
    }
  ],
  title: 'Company Revenue and Profit'
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
```



```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Data label](#)
- [Tooltip](#)

Histogram Chart in `##Platform_Name##` control

Histogram

Histogram type charts can provide a visual display of large amounts of data that are difficult to understand in a tabular or spreadsheet form. To render a histogram chart, use series [type](#) as `Histogram` and inject `HistogramSeries` module using `Chart.Inject(HistogramSeries)` method.

INDEX.JS

```

var chartData = [];
var points = [5.250, 7.750, 0, 8.275, 9.750, 7.750, 8.275, 6.250,
5.750,
5.250, 23.000, 26.500, 27.750, 25.025, 26.500, 26.500, 28.025,
29.250, 26.750, 27.250,
26.250, 25.250, 34.500, 25.625, 25.500, 26.625, 36.275, 36.250,
26.875, 40.000, 43.000,
46.500, 47.750, 45.025, 56.500, 56.500, 58.025, 59.250, 56.750,
57.250,
46.250, 55.250, 44.500, 45.525, 55.500, 46.625, 46.275, 56.250,
46.875, 43.000,
46.250, 55.250, 44.500, 45.425, 55.500, 56.625, 46.275, 56.250,
46.875, 43.000,
46.250, 55.250, 44.500, 45.425, 55.500, 46.625, 56.275, 46.250,
56.875, 41.000, 63.000,
66.500, 67.750, 65.025, 66.500, 76.500, 78.025, 79.250, 76.750,
77.250,
66.250, 75.250, 74.500, 65.625, 75.500, 76.625, 76.275, 66.250,
66.875, 80.000, 85.250,
87.750, 89.000, 88.275, 89.750, 97.750, 98.275, 96.250, 95.750,
95.250

```

```

    ];
    points.map(function (value) {
        chartData.push({
            y: value
        });
    });
    var chart = new ej.charts.Chart({
        primaryXAxis: {
            minimum: 0, maximum: 100
        },
        legendSettings: { visible: false },
        primaryYAxis: {
            minimum: 0, maximum: 50, interval: 10,
        },
        series: [
            {
                type: 'Histogram', width: 2, yName: 'y', name: 'Score',
                dataSource: chartData, binInterval: 20,
                showNormalDistribution: true, columnWidth: 0.99
            }
        ],
    }, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Data label](#)
- [Tooltip](#)

Error bar Chart in `##Platform_Name##` control

Error Bar

Error bars are graphical representations of the variability of data and used on graphs to indicate the error or uncertainty in a reported measurement. To render the error bar for the series, set [visible](#) as `true` and inject `ErrorBar` module using `Chart.Inject(ErrorBar)` method.

INDEX.JS

```
var chartData = [
  { x: 2006, y: 7.8 }, { x: 2007, y: 7.2 },
  { x: 2008, y: 6.8 }, { x: 2009, y: 10.7 },
  { x: 2010, y: 10.8 }, { x: 2011, y: 9.8 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    minimum: 2005, maximum: 2012, interval: 1,
    title: 'Year'
  },
  primaryYAxis: {
    minimum: 3, maximum: 12,
    interval: 1, title: 'Percentage',
    labelFormat: '{value}%'
  },
  series: [{
    dataSource: chartData,
    xName: 'x', yName: 'y',
    errorBar: {
      visible: true,
    }, marker: {
      visible: true,
    }, animation: { enable: false },
    name: 'India',
    type: 'Line'
  }],
  title: 'Unemployment rate (%)',
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customizing Error Bar

To customize the error bar type, set error bar [type](#) as **Custom** and then change the horizontal/vertical positive and negative error of error bar.

INDEX.JS

```

var chartData = [
    { x: 2006, y: 7.8 }, { x: 2007, y: 7.2 },
    { x: 2008, y: 6.8 }, { x: 2009, y: 5.7 },
    { x: 2010, y: 8.8 }, { x: 2011, y: 9.8 }
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        minimum: 2005, maximum: 2012, interval: 1,
        title: 'Year'
    },
    primaryYAxis: {
        minimum: 3, maximum: 12,
        interval: 1, title: 'Percentage',
        labelFormat: '{value}%'
    },
    series: [{
        dataSource: chartData,
        xName: 'x', yName: 'y',
        errorBar: {
            visible: true,
            type: 'Custom',
            mode: 'Both',
            verticalPostiveError: 3,
            horizontalPositiveError: 2,
            verticalNegativeError: 3,
            horizontalNegativeError: 2
        }, marker: {
            visible: true,
        }, animation: { enable: false },
        name: 'India',
        type: 'Line'
    }],
    title: 'Unemployment rate (%)',

```

```
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Error Bar Mode

Error bar mode is used to define whether the error bar line has to be drawn horizontally, vertically or in both side. To change the error bar mode use [mode](#) option.

INDEX.JS

```
var chartData = [
  { x: 2006, y: 7.8 }, { x: 2007, y: 7.2 },
  { x: 2008, y: 6.8 }, { x: 2009, y: 5.7 },
  { x: 2010, y: 8.8 }, { x: 2011, y: 9.8 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    minimum: 2005, maximum: 2012, interval: 1,
    title: 'Year'
  },
  primaryYAxis: {
    minimum: 3, maximum: 12,
    interval: 1, title: 'Percentage',
    labelFormat: '{value}%'
  },
  series: [{
    dataSource: chartData,
```

```

        xName: 'x', yName: 'y',
        errorBar: {
            visible: true,
            mode: 'Horizontal'
        }, marker: {
            visible: true,
        }, animation: { enable: false },
        name: 'India',
        type: 'Line'
    }],
    title: 'Unemployment rate (%)',
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Error Bar Direction

To change the error bar direction to plus, minus or both side using [direction](#) option.

INDEX.JS

```

var chartData = [
  { x: 2006, y: 7.8 }, { x: 2007, y: 7.2 },
  { x: 2008, y: 6.8 }, { x: 2009, y: 10.7 },
  { x: 2010, y: 10.8 }, { x: 2011, y: 9.8 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    minimum: 2005, maximum: 2012, interval: 1,

```

```

        title: 'Year'
    },
    primaryYAxis: {
        minimum: 3, maximum: 12,
        interval: 1, title: 'Percentage',
        labelFormat: '{value}%'
    },
    series: [{
        dataSource: chartData,
        xName: 'x', yName: 'y',
        errorBar: {
            visible: true,
            mode: 'Vertical',
            direction: 'Minus'
        }, marker: {
            visible: true,
        }, animation: { enable: false },
        name: 'India',
        type: 'Line'
    }],
    title: 'Unemployment rate (%)',
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

Customizing Error Bar Cap

To customize the error bar cap length, width and fill color, you can use [errorBarCap](#) option.

INDEX.JS

```

var chartData = [
    { x: 2006, y: 7.8 }, { x: 2007, y: 7.2 },
    { x: 2008, y: 6.8 }, { x: 2009, y: 10.7 },
    { x: 2010, y: 10.8 }, { x: 2011, y: 9.8 }
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        minimum: 2005, maximum: 2012, interval: 1,
        title: 'Year'
    },
    primaryYAxis: {
        minimum: 3, maximum: 12,
        interval: 1, title: 'Percentage',
        labelFormat: '{value}%'
    },
    series: [{
        dataSource: chartData,
        xName: 'x', yName: 'y',
        errorBar: {
            visible: true,
            errorBarCap: {
                length: 10,
                width: 10,
                color: '#0000ff'
            }
        }, marker: {
            visible: true,
        }, animation: { enable: false },
        name: 'India',
        type: 'Line'
    }],
    title: 'Unemployment rate (%)',
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>

```



```

    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customizing Error Bar Color

To customise the error bar color for individual errors, use the `errorBarColorMapping` property. You can also customize the vertical error, horizontal error, horizontal negative and positive error and vertical negative and positive error for an individual point using [verticalError](#), [horizontalError](#), [horizontalNegativeError](#), [horizontalPositiveError](#), [verticalNegativeError](#) and [verticalPositiveError](#) properties.

INDEX.JS

```

var chartData = [
    { x: 2006, y: 7.8, color: 'red', error: 4 }, { x: 2007, y: 7.2, color:
'blue', error: 3 },
    { x: 2008, y: 6.8, color: 'green', error: 1 }, { x: 2009, y: 10.7,
color: 'orange', error: 5 },
    { x: 2010, y: 10.8, color: 'yellow', error: 7 }, { x: 2011, y: 9.8,
color: 'grey', error: 2 }
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        minimum: 2005, maximum: 2012, interval: 1,
        title: 'Year'
    },
    primaryYAxis: {
        minimum: 3, maximum: 12,
        interval: 1, title: 'Percentage',
        labelFormat: '{value}%'
    },
    series: [{
        dataSource: chartData,
        xName: 'x', yName: 'y',
        errorBar: {
            visible: true,
            errorBarColorMapping: 'color',
            verticalError: 'error'
        }, marker: {
            visible: true,
        }, animation: { enable: false },
        name: 'India',
        type: 'Line'
    }],
    title: 'Unemployment rate (%)',
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Data label](#)
- [Tooltip](#)

Vertical Chart in ##Platform_Name## control

Vertical chart

In EJ2 chart, you can draw a chart in vertical manner by changing orientation of the axis. All series types support this feature.

You can use `isTransposed` property in chart to render a chart in vertical manner.

INDEX.JS

```

var chartData = [
  { x: 'Jan', y: -1 }, { x: 'Feb', y: -1 }, { x: 'Mar', y: 2 },
  { x: 'Apr', y: 8 }, { x: 'May', y: 13 }, { x: 'Jun', y: 18 },
  { x: 'Jul', y: 21 }, { x: 'Aug', y: 20 }, { x: 'Sep', y: 16 },
  { x: 'Oct', y: 10 }, { x: 'Nov', y: 4 }, { x: 'Dec', y: 0 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    title: 'Month',
    valueType: 'Category'
  },
  primaryYAxis: {
    minimum: -5, maximum: 25, interval: 10,

```

```

        labelFormat: '{value}°C',
        majorGridLines: { width : 0}
    },
    series:[{
        dataSource: chartData, width:2,
        xName: 'x', yName: 'y',
        name: 'London',
        // Series type as spline series
        type: 'Spline',
        marker: { visible: true}
    }],
    isTransposed: true,
    title: 'Climate Graph-2012'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Data label](#)
- [Tooltip](#)

Pare to Chart in `##Platform_Name##` control

Pareto

Pareto charts are used to find the cumulative values of data in different categories. It is a combination of Column and Line series.

The initial values are represented by column chart and the cumulative values are represented by Line chart. To render a pareto chart, use series `type` as Pareto and inject `ParetoSeries` `ColumnSeries` and `LineSeries` module using `Chart.Inject(ParetoSeries, LineSeries, ColumnSeries)` method.

INDEX.JS

```
/**
 * Sample for Pareto chart
 */
var chart = new ej.charts.Chart({
  //Initializing Primary X Axis
  primaryXAxis: {
    interval: 1,
    valueType: 'Category', labelIntersectAction: 'Rotate45',
    majorGridLines: { width: 0 }, minorGridLines: { width: 0 },
    majorTickLines: { width: 0 }, minorTickLines: { width: 0 },
    lineStyle: { width: 0 },
  },
  //Initializing Primary Y Axis
  primaryYAxis: {
    title: 'Frequency of Occurence',
    minimum: 0,
    maximum: 25,
    interval: 5,
    lineStyle: { width: 0 },
    majorTickLines: { width: 0 }, majorGridLines: { width: 1 },
    minorGridLines: { width: 1 }, minorTickLines: { width: 0 }
  },
  chartArea: {
    border: {
      width: 0
    }
  },
  //Initializing Chart Series
  series: [
    {
      type: 'Pareto',
      dataSource: [
        { x: 'Button Defect', y: 23 }, { x: 'Pocket Defect', y: 16 },
        { x: 'Collar Defect', y: 10 }, { x: 'Cuff Defect', y: 7 },
        { x: 'Sleeve Defect', y: 6 }
      ], marker: { visible: true },
      xName: 'x', yName: 'y', name: 'Defect', width: 2
    }
  ],
  width: ej.base.Browser.isDevice ? '100%' : '75%',
  //Initializing Chart Title
  title: 'Pareto chart - Defects in Shirts',
  legendSettings: { visible: false },
});
```

```
//Initializing Tooltip
tooltip: {
    enable: true,
    shared: false
},
});
chart.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Pareto customization

The pareto line series can be customized by using the [marker](#), [width](#), [dashArray](#), and [fill](#) properties in the [paretoOptions](#). The secondary axis for the pareto series can be shown or hidden using the [showAxis](#) property.

INDEX.JS

```
/**
 * Sample for Pareto chart
 */
var chart = new ej.charts.Chart({
  //Initializing Primary X Axis
  primaryXAxis: {
    interval: 1,
    valueType: 'Category', labelIntersectAction: 'Rotate45',
    majorGridLines: { width: 0 }, minorGridLines: { width: 0 },
    majorTickLines: { width: 0 }, minorTickLines: { width: 0 },
```

```

        lineStyle: { width: 0 },
    },
    //Initializing Primary X Axis
    primaryYAxis: {
        title: 'Frequency of Occurence',
        minimum: 0,
        maximum: 25,
        interval: 5,
        lineStyle: { width: 0 },
        majorTickLines: { width: 0 }, majorGridLines: { width: 1 },
        minorGridLines: { width: 1 }, minorTickLines: { width: 0 }
    },
    chartArea: {
        border: {
            width: 0
        }
    },
    //Initializing Chart Series
    series: [
        {
            type: 'Pareto',
            dataSource: [
                { x: 'Button Defect', y: 23 }, { x: 'Pocket Defect', y: 16 },
                { x: 'Collar Defect', y: 10 }, { x: 'Cuff Defect', y: 7 },
                { x: 'Sleeve Defect', y: 6 }, { x: 'Other Defect', y: 2 }
            ],
            xName: 'x', yName: 'y', name: 'Defect', width: 2, opacity: 0.75,
            columnWidth: 0.4,
            paretoOptions: {
                marker: { visible: true, isFilled: true, width: 7, height: 7 },
                dashArray: '3,2',
                width: 2,
                fill: '#F7523F'
            },
            cornerRadius: { topLeft: ej.base.Browser.isDevice ? 4 : 6,
            topRight: ej.base.Browser.isDevice ? 4 : 6 }
        },
        {
            //Initializing Chart Title
            title: 'Defects in Shirts',
            legendSettings: { visible: true, enableHighlight: true },
            //Initializing Tooltip
            tooltip: {
                enable: true,
                shared: true,
                format: '${series.name} : <b>${point.y}</b>'
            }
        }
    ]
});
chart.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>

```

```

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See also

- [Data label](#)
- [Tooltip](#)

<!-- markdownlint-disable MD036 -->

Polar radar in ##Platform_Name## Chart control

Polar Chart

To render a polar series, use series [type](#) as **Polar** and inject **PolarSeries** module using **Chart.Inject(PolarSeries)** method.

Draw Types

Polar drawType property is used to change the series plotting type to line, column, area, range column, spline, scatter, stacking area and stacking column. The default value of drawType is **Line**.

Line

To render a line draw type, use series [drawType](#) as **Line** and inject **LineSeries** module using **Chart.Inject(LineSeries)** method.

[isClosed](#) property specifies whether to join start and end point of a line series used in polar chart to form a closed path. Default value of isClosed is true.

INDEX.JS

```
var chartData = [
```

```

    { x: 2005, y: 28 }, { x: 2006, y: 25 }, { x: 2007, y: 26 }, { x: 2008, y:
27 },
    { x: 2009, y: 32 }, { x: 2010, y: 35 }, { x: 2011, y: 30 }
  ];
  var chart = new ej.charts.Chart({
    primaryXAxis: {
      title: 'Year',
      minimum: 2004, maximum: 2012, interval: 1
    },
    primaryYAxis: {
      minimum: 20, maximum: 40, interval: 5,
      title: 'Efficiency',
      labelFormat: '{value}%'
    },
    series:[{
      dataSource: chartData, width:2,
      xName: 'x', yName: 'y',
      name: 'India',
      //Series type as polar
      type: 'Polar',
      // Series draw type as line
      drawType: 'Line'
    }],
    title: 'Efficiency of oil-fired power production'
  }, '#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
  var ele = document.getElementById('container');
  if(ele) {
    ele.style.visibility = "visible";
  }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```


Spline

To render a spline draw type, use series [drawType](#) as `Spline` and inject `SplineSeries` module using `Chart.Inject(SplineSeries)` method.

INDEX.JS

```
var chartData = [
  { x: 'Jan', y: -1 }, { x: 'Feb', y: -1 }, { x: 'Mar', y: 2 },
  { x: 'Apr', y: 8 }, { x: 'May', y: 13 }, { x: 'Jun', y: 18 },
  { x: 'Jul', y: 21 }, { x: 'Aug', y: 20 }, { x: 'Sep', y: 16 },
  { x: 'Oct', y: 10 }, { x: 'Nov', y: 4 }, { x: 'Dec', y: 0 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    title: 'Month',
    valueType: 'Category'
  },
  primaryYAxis: {
    minimum: -5, maximum: 35, interval: 10,
    title: 'Temperature in Celsius',
    labelFormat: '{value}C'
  },
  series:[{
    dataSource: chartData, width:2,
    xName: 'x', yName: 'y',
    name: 'London',
    // Series type as Polar series
    type: 'Polar',
    // Series draw type as spline
    drawType: 'Spline'
  }],
  title: 'Climate Graph-2012'
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
</body>
</html>
```

```

var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Area

To render a area draw type, use series [drawType](#) as `Area` and inject `AreaSeries` module using `Chart.Inject(AreaSeries)` method.

INDEX.JS

```

var chartData = [
    { x: 1900, y: 4 }, { x: 1920, y: 3.0 }, { x: 1940, y: 3.8 },
    { x: 1960, y: 3.4 }, { x: 1980, y: 3.2 }, { x: 2000, y: 3.9 }
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        title: 'Year',
        minimum: 1900, maximum: 2000, interval: 10,
        edgeLabelPlacement: 'Shift'
    },
    primaryYAxis: {
        minimum: 2, maximum: 5, interval: 0.5,
        title: 'Sales Amount in Millions'
    },
    series:[{
        dataSource: chartData,
        xName: 'x', yName: 'y',
        opacity: 0.5, fill: '#69D2E7',
        name: 'Product A',
        // Series type as polar series
        type: 'Polar',
        // Series draw type as area
        drawType: 'Area'
    }],
    title: 'Average Sales Comparison'
}, '#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>

```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Stacked Area

To render a stacked area draw type, use series [drawType](#) as `StackingArea` and inject `StackingAreaSeries` module using `Chart.Inject(StackingAreaSeries)` method.

INDEX.JS

```

var chartData = [
    { x: '2000', y: 0.61, y1: 0.03, y2: 0.48 },
    { x: '2001', y: 0.81, y1: 0.05, y2: 0.53 },
    { x: '2002', y: 0.91, y1: 0.06, y2: 0.57 },
    { x: '2003', y: 1, y1: 0.09, y2: 0.61 },
    { x: '2004', y: 1.19, y1: 0.14, y2: 0.63 },
    { x: '2005', y: 1.47, y1: 0.20, y2: 0.64 },
    { x: '2006', y: 1.74, y1: 0.29, y2: 0.66 },
    { x: '2007', y: 1.98, y1: 0.46, y2: 0.76 },
    { x: '2008', y: 1.99, y1: 0.64, y2: 0.77 },
    { x: '2009', y: 1.70, y1: 0.75, y2: 0.55 },
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        title: 'Years',
        valueType: 'Category',
        majorTickLines: { width: 0 },
        edgeLabelPlacement: 'Shift'
    },
    primaryYAxis: {
        {
            title: 'Spend in Billions',
            minimum: 0,
            maximum: 4,
            interval: 1,
            majorTickLines: { width: 0 },
            labelFormat: '{value}B'
        },
    },
    series: [
        {
            dataSource: chartData, xName: 'x', yName: 'y',
            // Series type as polar series

```

```

        type : 'Polar',
        //Series draw type as stacked area series
        drawType: 'StackingArea',
        name: 'Organic',
    }, {
        dataSource: chartData, xName: 'x', yName: 'y1',
        type : 'Polar',
        drawType: 'StackingArea', name: 'Fair-trade',
    }, {
        dataSource: chartData, xName: 'x', yName: 'y2',
        type : 'Polar',
        drawType: 'StackingArea', name: 'Veg Alternatives',
    },
    ],
    title: 'Trend in Sales of Ethical Produce'
}, '#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Column

To render a column draw type, use series [drawType](#) as **Column**.

INDEX.JS

```

var chartData = [
  { country: "USA", gold: 50 },
  { country: "China", gold: 40 },
```

```

        { country: "Japan", gold: 70 },
        { country: "Australia", gold: 60 },
        { country: "France", gold: 50 },
        { country: "Germany", gold: 40 },
        { country: "Italy", gold: 40 },
        { country: "Sweden", gold: 30 }
    ];
    var chart = new ej.charts.Chart({
        primaryXAxis: {
            valueType: 'Category',
            title: 'Countries'
        },
        primaryYAxis: {
            minimum: 0, maximum: 80,
            interval: 20, title: 'Medals'
        },
        series:[{
            dataSource: chartData,
            xName: 'country', yName: 'gold',
            name: 'Gold',
            // Series type as polar series
            type: 'Polar',
            // Series draw type as column series
            drawType: 'Column'
        }],
        title: 'Olympic Medals'
    }, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
    <script>
    var ele = document.getElementById('container');
    if(ele) {
        ele.style.visibility = "visible";
    }
    </script>
    <script src="index.js" type="text/javascript"></script>

```

```
</body></html>
```

Stacked Column

To render a stacked column draw type, use series [drawType](#) as `StackingColumn`.

INDEX.JS

```
var chartData = [
  { x: '2014', y: 111.1, y1: 76.9, y2: 66.1, y3: 34.1 },
  { x: '2015', y: 127.3, y1: 99.5, y2: 79.3, y3: 38.2 },
  { x: '2016', y: 143.4, y1: 121.7, y2: 91.3, y3: 44.0 },
  { x: '2017', y: 159.9, y1: 142.5, y2: 102.4, y3: 51.6 },
  { x: '2018', y: 175.4, y1: 166.7, y2: 112.9, y3: 61.9 },
  { x: '2019', y: 189.0, y1: 182.9, y2: 122.4, y3: 71.5 },
  { x: '2020', y: 202.7, y1: 197.3, y2: 120.9, y3: 82.0 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    title: 'Years',
    interval: 1,
    valueType: 'Category'
  },
  primaryYAxis: {
    title: 'Sales in Billions',
    minimum: 0,
    maximum: 700,
    interval: 100,
    labelFormat: '{value}B',
  },
  series: [
    {
      dataSource: chartData, xName: 'x', yName: 'y',
      type: 'Polar',
      //Series draw type as stacked column
      drawType: 'StackingColumn',
      name: 'UK',
    }, {
      dataSource: chartData, xName: 'x', yName: 'y1',
      type: 'Polar',
      drawType: 'StackingColumn', name: 'Germany',
    }, {
      dataSource: chartData, xName: 'x', yName: 'y2',
      type: 'Polar',
      drawType: 'StackingColumn', name: 'France',
    }, {
      dataSource: chartData, xName: 'x', yName: 'y3',
      type: 'Polar',
      drawType: 'StackingColumn', name: 'Italy',
    }
  ],
  title: 'Mobile Game Market by Country'
}, '#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Range Column

To render a range column draw type, use series [drawType](#) as **RangeColumn**.

INDEX.JS

```

var data = [
  { x: 'Jan', low: 0.7, high: 6.1 }, { x: 'Feb', low: 1.3, high: 6.3 }, {
x: 'Mar', low: 1.9, high: 8.5 },
  { x: 'Apr', low: 3.1, high: 10.8 }, { x: 'May', low: 5.7, high: 14.40 },
{ x: 'Jun', low: 8.4, high: 16.90 },
  { x: 'Jul', low: 10.6,high: 19.20 }, { x: 'Aug', low: 10.5,high: 18.9 },
{ x: 'Sep', low: 8.5, high: 16.1 },
  { x: 'Oct', low: 6.0, high: 12.5 }, { x: 'Nov', low: 1.5, high: 6.9 },
{ x: 'Dec', low: 5.1, high: 12.1 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    title: 'Months',
    valueType: 'Category'
  },
  primaryYAxis:
  {
    title: 'Temperature(Celsius)',
  },
  series: [
    {
      type: 'Polar',
      // Series draw type as range column series

```

```

        drawType: 'RangeColumn',
        name: 'India',
        dataSource: data, xName: 'x', high: 'high', low: 'low',
    },
],
title: 'Maximum and Minimum Temperature'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Scatter

To render a scatter draw type, use series [DrawType](#) as `Scatter` and inject `ScatterSeries` module using `Chart.Inject(ScatterSeries)` method.

INDEX.JS

```

var chartData = [
  { x: 'Jan', y: -1 }, { x: 'Feb', y: -1 }, { x: 'Mar', y: 2 },
  { x: 'Apr', y: 8 }, { x: 'May', y: 13 }, { x: 'Jun', y: 18 },
  { x: 'Jul', y: 21 }, { x: 'Aug', y: 20 }, { x: 'Sep', y: 16 },
  { x: 'Oct', y: 10 }, { x: 'Nov', y: 4 }, { x: 'Dec', y: 0 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    title: 'Month',
    valueType: 'Category'
  },

```



```

primaryYAxis: {
    minimum: -5, maximum: 35, interval: 10,
    title: 'Temperature in Celsius',
    labelFormat: '{value}C'
}
series:[{
    dataSource: chartData, width:2,
    xName: 'x', yName: 'y',
    name: 'London',
    // Series type as Polar series
    type: 'Polar'
    // Series draw type as scatter
    drawType: 'Scatter'
}],
title: 'Climate Graph-2012'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Radar Chart

To render a radar series, use series [type](#) as **Radar** and inject **PolarSeries** module using **Chart.Inject(RadarSeries)** method.

Draw Type

similar to Polar drawType, Radar draw property is used to change the series plotting type to line, column, area, range column, spline, scatter, stacking area and stacking column. The default value of drawType is Line.

INDEX.JS

```
var chartData = [
  { x: 2005, y: 28 }, { x: 2006, y: 25 }, { x: 2007, y: 26 }, { x: 2008, y:
27 },
  { x: 2009, y: 32 }, { x: 2010, y: 35 }, { x: 2011, y: 30 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    title: 'Year',
    minimum: 2004, maximum: 2012, interval: 1
  },
  primaryYAxis: {
    minimum: 20, maximum: 40, interval: 5,
    title: 'Efficiency',
    labelFormat: '{value}%'
  },
  series:[{
    dataSource: chartData, width:2,
    xName: 'x', yName: 'y',
    name: 'India',
    //Series type as polar
    type: 'Radar',
    // Series draw type as line
    drawType: 'Line'
  }],
  title: 'Efficiency of oil-fired power production'
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
</body>
```

```

var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customization

Start Angle

You can customize the start angle of the polar series using [startAngle](#) property. By default, `startAngle` is 0 degree.

INDEX.JS

```

var chartData = [
    { country: "USA", gold: 50 },
    { country: "China", gold: 40 },
    { country: "Japan", gold: 70 },
    { country: "Australia", gold: 60 },
    { country: "France", gold: 50 },
    { country: "Germany", gold: 40 },
    { country: "Italy", gold: 40 },
    { country: "Sweden", gold: 30 }
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        valueType: 'Category',
        title: 'Countries',
        startAngle: 90,
    },
    primaryYAxis: {
        minimum: 0, maximum: 80,
        interval: 20, title: 'Medals'
    },
    series:[{
        dataSource: chartData,
        xName: 'country', yName: 'gold',
        name: 'Gold',
        // Series type as polar series
        type: 'Polar',
        // Series draw type as column series
        drawType: 'Column'
    }],
    title: 'Olympic Medals'
}, '#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">

```

```

<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Coefficient in axis

You can customize the radius of the polar series using [coefficient](#) property. By default, `coefficient` is 100.

INDEX.JS

```

var chartData = [
    { country: "USA", gold: 50 },
    { country: "China", gold: 40 },
    { country: "Japan", gold: 70 },
    { country: "Australia", gold: 60 },
    { country: "France", gold: 50 },
    { country: "Germany", gold: 40 },
    { country: "Italy", gold: 40 },
    { country: "Sweden", gold: 30 }
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        valueType: 'Category',
        title: 'Countries',
        coefficient: 80
    },
    primaryYAxis: {
        minimum: 0, maximum: 80,
        interval: 20, title: 'Medals'
    },
    series:[{
        dataSource: chartData,
        xName: 'country', yName: 'gold',
        name: 'Gold',
        // Series type as polar series
        type: 'Polar',
        // Series draw type as column series
    }]
});

```

```

        drawType: 'Column'
    }],
    title: 'Olympic Medals'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Data label](#)
- [Tooltip](#)

Chart series in ##Platform_Name## Chart control

Multiple Series

You can add multiple series to the chart by using [series](#) property. The series are rendered in the order as it is added to the series array.

INDEX.JS

```

var chartData = [
  { country: "USA", gold: 50, silver: 70, bronze: 45 },
  { country: "China", gold: 40, silver: 60, bronze: 55 },
  { country: "Japan", gold: 70, silver: 60, bronze: 50 },
  { country: "Australia", gold: 60, silver: 56, bronze: 40 },
  { country: "France", gold: 50, silver: 45, bronze: 35 },
  { country: "Germany", gold: 40, silver: 30, bronze: 22 },

```

```

        { country: "Italy", gold: 40, silver: 35, bronze: 37 },
        { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
    ];
    var chart = new ej.charts.Chart({
        primaryXAxis: {
            valueType: 'Category',
            title: 'Countries'
        },
        primaryYAxis: {
            minimum: 0, maximum: 80,
            interval: 20, title: 'Medals'
        },
        series:[{
            dataSource: chartData,
            xName: 'country', yName: 'gold',
            name: 'Gold', type: 'Column'
        }, {
            dataSource: chartData,
            xName: 'country', yName: 'silver',
            name: 'Silver', type: 'Column'
        }, {
            dataSource: chartData,
            xName: 'country', yName: 'bronze',
            name: 'Bronze', type: 'Column'
        }
    ],
        title: 'Olympic Medals'
    }, '#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
    <script>
    var ele = document.getElementById('container');
    if(ele) {
        ele.style.visibility = "visible";
    }
    </script>
    <script src="index.js" type="text/javascript"></script>
```

```
</body></html>
```

Combination Series

Bar series cannot be combined with any other series as the axis orientation is different from other series.

INDEX.JS

```
var chartData = [
  { x: '2005', y: 1.2, y1: 0.5, y2: 0.7, y3: -0.8, y4: 1.5 },
  { x: '2006', y: 1, y1: 0.5, y2: 1.4, y3: 0, y4: 2.3 },
  { x: '2007', y: 1, y1: 0.5, y2: 1.5, y3: -1, y4: 2 },
  { x: '2008', y: 0.25, y1: 0.35, y2: 0.35, y3: -.35, y4: 0.1 },
  { x: '2009', y: 0.1, y1: 0.9, y2: -2.7, y3: -0.3, y4: -2.7 },
  { x: '2010', y: 1, y1: 0.5, y2: 0.5, y3: -0.5, y4: 1.8 },
  { x: '2011', y: 0.1, y1: 0.25, y2: 0.25, y3: 0, y4: 2 },
  { x: '2012', y: -0.25, y1: -0.5, y2: -0.1, y3: -0.4, y4: 0.4 },
  { x: '2013', y: 0.25, y1: 0.5, y2: -0.3, y3: 0, y4: 0.9 },
  { x: '2014', y: 0.6, y1: 0.6, y2: -0.6, y3: -0.6, y4: 0.4 },
  { x: '2015', y: 0.9, y1: 0.5, y2: 0, y3: -0.3, y4: 1.3 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    title: 'Years',
    interval: 1,
    labelIntersectAction: 'Rotate45',
    valueType: 'Category'
  },
  primaryYAxis: {
    {
      title: 'Growth',
      minimum: -3, maximum: 3, interval: 1
    }
  },
  series: [
    {
      type: 'StackingColumn', dataSource: chartData,
      xName: 'x', yName: 'y', name: 'Private Consumption'
    }, {
      type: 'StackingColumn', dataSource: chartData,
      xName: 'x', yName: 'y1', name: 'Government Consumption'
    }, {
      type: 'StackingColumn', dataSource: chartData,
      xName: 'x', yName: 'y2', name: 'Investment'
    }, {
      type: 'StackingColumn', dataSource: chartData,
      xName: 'x', yName: 'y3', name: 'Net Foreign Trade'
    }, {
      type: 'Line', name: 'GDP',
      dataSource: chartData, xName: 'x', yName: 'y4',
      width: 2, opacity: 0.6,
      marker: {
        visible: true,
        width: 10, opacity: 0.6,
        height: 10
      }
    }
  ]
});
```

```

    ],
    title: 'Annual Growth GDP in France'
  }, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Enable Complex Property in Series

By setting `enableComplexProperty` value as `true` in series you can bind complex data to the chart.

INDEX.JS

```

var chartData = [
  { x: '2005', y: 1.2, y1: 0.5, y2: 0.7, y3: -0.8, y4: 1.5 },
  { x: '2006', y: 1, y1: 0.5, y2: 1.4, y3: 0, y4: 2.3 },
  { x: '2007', y: 1, y1: 0.5, y2: 1.5, y3: -1, y4: 2 },
  { x: '2008', y: 0.25, y1: 0.35, y2: 0.35, y3: -.35, y4: 0.1 },
  { x: '2009', y: 0.1, y1: 0.9, y2: -2.7, y3: -0.3, y4: -2.7 },
  { x: '2010', y: 1, y1: 0.5, y2: 0.5, y3: -0.5, y4: 1.8 },
  { x: '2011', y: 0.1, y1: 0.25, y2: 0.25, y3: 0, y4: 2 },
  { x: '2012', y: -0.25, y1: -0.5, y2: -0.1, y3: -0.4, y4: 0.4 },
  { x: '2013', y: 0.25, y1: 0.5, y2: -0.3, y3: 0, y4: 0.9 },
  { x: '2014', y: 0.6, y1: 0.6, y2: -0.6, y3: -0.6, y4: 0.4 },
  { x: '2015', y: 0.9, y1: 0.5, y2: 0, y3: -0.3, y4: 1.3 }
];

var chart = new ej.charts.Chart({
  primaryXAxis: {
    title: 'Years',

```



```

        interval: 1,
        labelIntersectAction : 'Rotate45',
        valueType: 'Category'
    },
    primaryYAxis:
    {
        title: 'Growth',
        minimum: -3, maximum: 3, interval: 1
    },
    series: [
        {
            type: 'StackingColumn', dataSource: chartData,
            xName: 'x', yName: 'y', name: 'Private Consumption'
        }, {
            type: 'StackingColumn', dataSource: chartData,
            xName: 'x', yName: 'y1', name: 'Government Consumption'
        }, {
            type: 'StackingColumn', dataSource: chartData,
            xName: 'x', yName: 'y2', name: 'Investment'
        }, {
            type: 'StackingColumn', dataSource: chartData,
            xName: 'x', yName: 'y3', name: 'Net Foreign Trade'
        }, {
            type: 'Line', name: 'GDP',
            dataSource: chartData, xName: 'x', yName: 'y4',
            width: 2, opacity: 0.6,
            marker: {
                visible: true,
                width: 10, opacity: 0.6,
                height: 10
            },
        },
    ],
    title: 'Annual Growth GDP in France'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>

```

```

</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

<!-- markdownlint-disable MD036 -->

Technical indicators in ##Platform_Name## Chart control

A technical indicator is a mathematical calculation based on historic price, volume or open interest information that aims to forecast financial market direction.

Chart supports 10 types of technical indicators.

Accumulation Distribution

Accumulation Distribution combines price and volume to show how money may be flowing into or out of stock. To render a Accumulation Distribution Indicator, use indicator [type](#) as `AccumulationDistribution` and inject

`AccumulationDistributionIndicator` module using `Chart.Inject(AccumulationDistributionIndicator)`. To calculate the signal line [volume](#) field is additionally added with `dataSource`.

INDEX.JS

```

var chartData = [
    { x: new Date('2012-10-15'), open: 90.3357, high: 93.2557, low: 87.0885,
      close: 87.12, volume: 646996264 },
    { x: new Date('2012-10-22'), open: 87.4885, high: 90.7685, low: 84.4285,
      close: 86.2857, volume: 866040680 },
    { x: new Date('2012-10-29'), open: 84.9828, high: 86.1428, low: 82.1071,
      close: 82.4, volume: 367371310 },
    { x: new Date('2012-11-05'), open: 83.3593, high: 84.3914, low: 76.2457,
      close: 78.1514, volume: 919719846 },
    { x: new Date('2012-11-12'), open: 79.1643, high: 79.2143, low: 72.25,
      close: 75.3825, volume: 894382149 },
    { x: new Date('2012-11-19'), open: 77.2443, high: 81.7143, low: 77.1257,
      close: 81.6428, volume: 527416747 },
    { x: new Date('2012-11-26'), open: 82.2714, high: 84.8928, low: 81.7514,
      close: 83.6114, volume: 646467974 },
    { x: new Date('2012-12-03'), open: 84.8071, high: 84.9414, low: 74.09,
      close: 76.1785, volume: 980096264 },
    { x: new Date('2012-12-10'), open: 75, high: 78.5085, low: 72.2257,
      close: 72.8277, volume: 835016110 },
    { x: new Date('2012-12-17'), open: 72.7043, high: 76.4143, low: 71.6043,
      close: 74.19, volume: 726150329 },
    { x: new Date('2012-12-24'), open: 74.3357, high: 74.8928, low: 72.0943,
      close: 72.7984, volume: 321104733 },
    { x: new Date('2012-12-31'), open: 72.9328, high: 79.2857, low: 72.7143,
      close: 75.2857, volume: 540854882 },

```

```

{ x: new Date('2013-01-07'), open: 74.5714, high: 75.9843, low: 73.6,
close: 74.3285, volume: 574594262 },
{ x: new Date('2013-01-14'), open: 71.8114, high: 72.9643, low: 69.0543,
close: 71.4285, volume: 803105621 },
{ x: new Date('2013-01-21'), open: 72.08, high: 73.57, low: 62.1428,
close: 62.84, volume: 971912560 },
{ x: new Date('2013-01-28'), open: 62.5464, high: 66.0857, low: 62.2657,
close: 64.8028, volume: 656549587 },
{ x: new Date('2013-02-04'), open: 64.8443, high: 68.4014, low: 63.1428,
close: 67.8543, volume: 743778993 },
{ x: new Date('2013-02-11'), open: 68.0714, high: 69.2771, low: 65.7028,
close: 65.7371, volume: 585292366 },
{ x: new Date('2013-02-18'), open: 65.8714, high: 66.1043, low: 63.26,
close: 64.4014, volume: 421766997 },
{ x: new Date('2013-02-25'), open: 64.8357, high: 65.0171, low: 61.4257,
close: 61.4957, volume: 582741215 },
{ x: new Date('2013-03-04'), open: 61.1143, high: 62.2043, low: 59.8571,
close: 61.6743, volume: 632856539 },
{ x: new Date('2013-03-11'), open: 61.3928, high: 63.4614, low: 60.7343,
close: 63.38, volume: 572066981 },
{ x: new Date('2013-03-18'), open: 63.0643, high: 66.0143, low: 63.0286,
close: 65.9871, volume: 552156035 },
{ x: new Date('2013-03-25'), open: 66.3843, high: 67.1357, low: 63.0886,
close: 63.2371, volume: 390762517 },
{ x: new Date('2013-04-01'), open: 63.1286, high: 63.3854, low: 59.9543,
close: 60.4571, volume: 505273732 },
{ x: new Date('2013-04-08'), open: 60.6928, high: 62.57, low: 60.3557,
close: 61.4, volume: 387323550 },
{ x: new Date('2013-04-15'), open: 61, high: 61.1271, low: 55.0143,
close: 55.79, volume: 709945604 },
{ x: new Date('2013-04-22'), open: 56.0914, high: 59.8241, low: 55.8964,
close: 59.6007, volume: 787007506 },
{ x: new Date('2013-04-29'), open: 60.0643, high: 64.7471, low: 60,
close: 64.2828, volume: 655020017 },
{ x: new Date('2013-05-06'), open: 65.1014, high: 66.5357, low: 64.3543,
close: 64.71, volume: 545488533 },
{ x: new Date('2013-05-13'), open: 64.5014, high: 65.4143, low: 59.8428,
close: 61.8943, volume: 633706550 },
{ x: new Date('2013-05-20'), open: 61.7014, high: 64.05, low: 61.4428,
close: 63.5928, volume: 494379068 },
{ x: new Date('2013-05-27'), open: 64.2714, high: 65.3, low: 62.7714,
close: 64.2478, volume: 362907830 },
{ x: new Date('2013-06-03'), open: 64.39, high: 64.9186, low: 61.8243,
close: 63.1158, volume: 443249793 },
{ x: new Date('2013-06-10'), open: 63.5328, high: 64.1541, low: 61.2143,
close: 61.4357, volume: 389680092 },
{ x: new Date('2013-06-17'), open: 61.6343, high: 62.2428, low: 58.3,
close: 59.0714, volume: 400384818 },
{ x: new Date('2013-06-24'), open: 58.2, high: 58.38, low: 55.5528,
close: 56.6471, volume: 519314826 },
{ x: new Date('2013-07-01'), open: 57.5271, high: 60.47, low: 57.3171,
close: 59.6314, volume: 343878841 },
{ x: new Date('2013-07-08'), open: 60.0157, high: 61.3986, low: 58.6257,
close: 60.93, volume: 384106977 },
{ x: new Date('2013-07-15'), open: 60.7157, high: 62.1243, low: 60.5957,
close: 60.7071, volume: 286035513 },

```

```

    { x: new Date('2013-07-22'), open: 61.3514, high: 63.5128, low: 59.8157,
      close: 62.9986, volume: 395816827 },
    { x: new Date('2013-07-29'), open: 62.9714, high: 66.1214, low: 62.8857,
      close: 66.0771, volume: 339668858 },
    { x: new Date('2013-08-12'), open: 65.2657, high: 72.0357, low: 65.2328,
      close: 71.7614, volume: 711563584 },
    { x: new Date('2013-08-19'), open: 72.0485, high: 73.3914, low: 71.1714,
      close: 71.5743, volume: 417119660 },
    { x: new Date('2013-08-26'), open: 71.5357, high: 72.8857, low: 69.4286,
      close: 69.6023, volume: 392805888 },
    { x: new Date('2013-09-02'), open: 70.4428, high: 71.7485, low: 69.6214,
      close: 71.1743, volume: 317244380 },
    { x: new Date('2013-09-09'), open: 72.1428, high: 72.56, low: 66.3857,
      close: 66.4143, volume: 669376320 },
    { x: new Date('2013-09-16'), open: 65.8571, high: 68.3643, low: 63.8886,
      close: 66.7728, volume: 625142677 },
    { x: new Date('2013-09-23'), open: 70.8714, high: 70.9871, low: 68.6743,
      close: 68.9643, volume: 475274537 },
    { x: new Date('2013-09-30'), open: 68.1786, high: 70.3357, low: 67.773,
      close: 69.0043, volume: 368198906 },
    { x: new Date('2013-10-07'), open: 69.5086, high: 70.5486, low: 68.3257,
      close: 70.4017, volume: 361437661 },
    { x: new Date('2013-10-14'), open: 69.9757, high: 72.7514, low: 69.9071,
      close: 72.6985, volume: 342694379 },
    { x: new Date('2013-10-21'), open: 73.11, high: 76.1757, low: 72.5757,
      close: 75.1368, volume: 490458997 },
    { x: new Date('2013-10-28'), open: 75.5771, high: 77.0357, low: 73.5057,
      close: 74.29, volume: 508130174 },
    { x: new Date('2013-11-04'), open: 74.4428, high: 75.555, low: 73.1971,
      close: 74.3657, volume: 318132218 },
    { x: new Date('2013-11-11'), open: 74.2843, high: 75.6114, low: 73.4871,
      close: 74.9987, volume: 306711021 },
    { x: new Date('2013-11-18'), open: 74.9985, high: 75.3128, low: 73.3814,
      close: 74.2571, volume: 282778778 },
  ];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    title: 'Months',
    valueType: 'DateTime',
    intervalType: 'Months',
    majorGridLines: { width: 0 },
    crosshairTooltip: { enable: true },
  },
  primaryYAxis: {
    title: 'Price (Million Dollars)',
    minimum: 30,
    maximum: 180,
    interval: 30,
  },
  axes: [{
    name: 'secondary',
    minimum: -7000000000, maximum: 5000000000,
    interval: 6000000000,
    majorGridLines: { width: 0 },
    opposedPosition: true
  }],
  series: [{

```

```

        dataSource: chartData, width: 2,
        xName: 'x', yName: 'y', low: 'low', high: 'high', close: 'close',
        volume: 'volume', open: 'open',
        name: 'Apple Inc',
        type: 'Candle', animation: { enable: true }
    }],
    indicators: [{
        type: 'AccumulationDistribution', field: 'Close', seriesName: 'Apple
Inc', yAxisName: 'secondary', fill: 'blue',
        period: 3, animation: { enable: true }
    }],
    tooltip: { enable: true, shared: true },
    chartArea: { border: { width: 0 } },
    axisLabelRender: (args) => {
        if (args.axis.name === 'secondary') {
            let value = (args.text) / 1000000000;
            args.text = (value) + 'bn';
        }
    },
    crosshair: { enable: true, lineType: 'Vertical' },
    title: 'AAPL - 2016/2017'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Average True Range (ATR)

ATR measures the stock volatility by comparing the current value with the previous value. To render a Average True Range (ATR) Indicator,

use indicator [type](#) as `Atr` and inject `AtrIndicator` module using `Chart.Inject(AtrIndicator)`.

INDEX.JS

```
var chartData = [
  {x: new Date('2012-10-15'), open: 90.3357, high: 93.2557, low: 87.0885, close: 87.12, volume: 646996264},
  {x: new Date('2012-10-22'), open: 87.4885, high: 90.7685, low: 84.4285, close: 86.2857, volume: 866040680 },
  {x: new Date('2012-10-29'), open: 84.9828, high: 86.1428, low: 82.1071, close: 82.4, volume: 367371310},
  {x: new Date('2012-11-05'), open: 83.3593, high: 84.3914, low: 76.2457, close: 78.1514, volume: 919719846},
  {x: new Date('2012-11-12'), open: 79.1643, high: 79.2143, low: 72.25, close: 75.3825, volume: 894382149},
  {x: new Date('2012-11-19'), open: 77.2443, high: 81.7143, low: 77.1257, close: 81.6428, volume: 527416747},
  {x: new Date('2012-11-26'), open: 82.2714, high: 84.8928, low: 81.7514, close: 83.6114, volume: 646467974},
  {x: new Date('2012-12-03'), open: 84.8071, high: 84.9414, low: 74.09, close: 76.1785, volume: 980096264},
  {x: new Date('2012-12-10'), open: 75, high: 78.5085, low: 72.2257, close: 72.8277, volume: 835016110},
  {x: new Date('2012-12-17'), open: 72.7043, high: 76.4143, low: 71.6043, close: 74.19, volume: 726150329},
  {x: new Date('2012-12-24'), open: 74.3357, high: 74.8928, low: 72.0943, close: 72.7984, volume: 321104733},
  {x: new Date('2012-12-31'), open: 72.9328, high: 79.2857, low: 72.7143, close: 75.2857, volume: 540854882},
  {x: new Date('2013-01-07'), open: 74.5714, high: 75.9843, low: 73.6, close: 74.3285, volume: 574594262},
  {x: new Date('2013-01-14'), open: 71.8114, high: 72.9643, low: 69.0543, close: 71.4285, volume: 803105621},
  {x: new Date('2013-01-21'), open: 72.08, high: 73.57, low: 62.1428, close: 62.84, volume: 971912560},
  {x: new Date('2013-01-28'), open: 62.5464, high: 66.0857, low: 62.2657, close: 64.8028, volume: 656549587},
  {x: new Date('2013-02-04'), open: 64.8443, high: 68.4014, low: 63.1428, close: 67.8543, volume: 743778993},
  {x: new Date('2013-02-11'), open: 68.0714, high: 69.2771, low: 65.7028, close: 65.7371, volume: 585292366},
  {x: new Date('2013-02-18'), open: 65.8714, high: 66.1043, low: 63.26, close: 64.4014, volume: 421766997},
  {x: new Date('2013-02-25'), open: 64.8357, high: 65.0171, low: 61.4257, close: 61.4957, volume: 582741215},
  {x: new Date('2013-03-04'), open: 61.1143, high: 62.2043, low: 59.8571, close: 61.6743, volume: 632856539},
  {x: new Date('2013-03-11'), open: 61.3928, high: 63.4614, low: 60.7343, close: 63.38, volume: 572066981},
  {x: new Date('2013-03-18'), open: 63.0643, high: 66.0143, low: 63.0286, close: 65.9871, volume: 552156035},
  {x: new Date('2013-03-25'), open: 66.3843, high: 67.1357, low: 63.0886, close: 63.2371, volume: 390762517},
```

```
{x: new Date('2013-04-01'), open: 63.1286, high: 63.3854, low: 59.9543, close: 60.4571, volume: 505273732},
{x: new Date('2013-04-08'), open: 60.6928, high: 62.57, low: 60.3557, close: 61.4, volume: 387323550},
{x: new Date('2013-04-15'), open: 61, high: 61.1271, low: 55.0143, close: 55.79, volume: 709945604},
{x: new Date('2013-04-22'), open: 56.0914, high: 59.8241, low: 55.8964, close: 59.6007, volume: 787007506},
{x: new Date('2013-04-29'), open: 60.0643, high: 64.7471, low: 60, close: 64.2828, volume: 655020017},
{x: new Date('2013-05-06'), open: 65.1014, high: 66.5357, low: 64.3543, close: 64.71, volume: 545488533},
{x: new Date('2013-05-13'), open: 64.5014, high: 65.4143, low: 59.8428, close: 61.8943, volume: 633706550},
{x: new Date('2013-05-20'), open: 61.7014, high: 64.05, low: 61.4428, close: 63.5928, volume: 494379068},
{x: new Date('2013-05-27'), open: 64.2714, high: 65.3, low: 62.7714, close: 64.2478, volume: 362907830},
{x: new Date('2013-06-03'), open: 64.39, high: 64.9186, low: 61.8243, close: 63.1158, volume: 443249793},
{x: new Date('2013-06-10'), open: 63.5328, high: 64.1541, low: 61.2143, close: 61.4357, volume: 389680092},
{x: new Date('2013-06-17'), open: 61.6343, high: 62.2428, low: 58.3, close: 59.0714, volume: 400384818},
{x: new Date('2013-06-24'), open: 58.2, high: 58.38, low: 55.5528, close: 56.6471, volume: 519314826},
{x: new Date('2013-07-01'), open: 57.5271, high: 60.47, low: 57.3171, close: 59.6314, volume: 343878841},
{x: new Date('2013-07-08'), open: 60.0157, high: 61.3986, low: 58.6257, close: 60.93, volume: 384106977},
{x: new Date('2013-07-15'), open: 60.7157, high: 62.1243, low: 60.5957, close: 60.7071, volume: 286035513},
{x: new Date('2013-07-22'), open: 61.3514, high: 63.5128, low: 59.8157, close: 62.9986, volume: 395816827},
{x: new Date('2013-07-29'), open: 62.9714, high: 66.1214, low: 62.8857, close: 66.0771, volume: 339668858},
{x: new Date('2013-08-12'), open: 65.2657, high: 72.0357, low: 65.2328, close: 71.7614, volume: 711563584},
{x: new Date('2013-08-19'), open: 72.0485, high: 73.3914, low: 71.1714, close: 71.5743, volume: 417119660},
{x: new Date('2013-08-26'), open: 71.5357, high: 72.8857, low: 69.4286, close: 69.6023, volume: 392805888},
{x: new Date('2013-09-02'), open: 70.4428, high: 71.7485, low: 69.6214, close: 71.1743, volume: 317244380},
{x: new Date('2013-09-09'), open: 72.1428, high: 72.56, low: 66.3857, close: 66.4143, volume: 669376320},
{x: new Date('2013-09-16'), open: 65.8571, high: 68.3643, low: 63.8886, close: 66.7728, volume: 625142677},
{x: new Date('2013-09-23'), open: 70.8714, high: 70.9871, low: 68.6743, close: 68.9643, volume: 475274537},
{x: new Date('2013-09-30'), open: 68.1786, high: 70.3357, low: 67.773, close: 69.0043, volume: 368198906},
{x: new Date('2013-10-07'), open: 69.5086, high: 70.5486, low: 68.3257, close: 70.4017, volume: 361437661},
{x: new Date('2013-10-14'), open: 69.9757, high: 72.7514, low: 69.9071, close: 72.6985, volume: 342694379},
```

```

    {x: new Date('2013-10-21'), open: 73.11, high: 76.1757, low:
72.5757,close: 75.1368,volume: 490458997},
    {x: new Date('2013-10-28'), open: 75.5771, high: 77.0357, low:
73.5057,close: 74.29,volume: 508130174},
    {x: new Date('2013-11-04'), open: 74.4428, high: 75.555, low:
73.1971,close: 74.3657,volume: 318132218},
    {x: new Date('2013-11-11'), open: 74.2843, high: 75.6114, low:
73.4871,close: 74.9987,volume: 306711021},
    {x: new Date('2013-11-18'), open: 74.9985, high: 75.3128, low:
73.3814,close: 74.2571,volume: 282778778},
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        title: 'Months',
        valueType: 'DateTime',
        intervalType: 'Months',
        majorGridLines: { width: 0},
        crosshairTooltip: { enable: true },
    },
    primaryYAxis: {
        title: 'Price (Million Dollars)',
        minimum: 30,
        maximum: 180,
        interval: 30,
    },
    axes: [{
        name: 'secondary',
        minimum:0,maximum:15,
        majorGridLines: { width: 0 },
        opposedPosition: true
    }],
    series:[{
        dataSource: chartData, width: 2,
        xName: 'x', yName: 'y', low: 'low', high: 'high', close : 'close',
volume: 'volume',open:'open',
        name: 'Apple Inc',
        //Series type as RangeColumn
        type: 'Candle', animation: { enable: true }
    }],
    indicators: [{
        type: 'Atr', field: 'Low', seriesName: 'Apple Inc', yAxisName:
'secondary', fill: 'blue',
        period: 3, animation: { enable: true}
    }],
    tooltip: { enable: true, shared: true },
    chartArea: { border: { width: 0 } },
    crosshair: { enable: true, lineType: 'Vertical' },
    title: 'AAPL - 2016/2017'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
<title>EJ2 Animation</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">

```



```

<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Bollinger Band

A chart overlay that shows the upper and lower limits of normal price movements based on the standard deviation of prices. To render a Bollinger Band, use indicator [type](#) as `BollingerBand` and inject `BollingerBands` module using `Chart.Inject(BollingerBands)` method. Bollinger band will be represented by three lines (upperLine, lowerLine, signalLine). The default values of the Bollinger Band [period](#) is 14 and [standardDeviations](#) is 2.

INDEX.JS

```

var chartData = [
    {x: new Date('2012-10-15'), open: 90.3357, high: 93.2557, low:
87.0885,close: 87.12,volume: 646996264},
    {x: new Date('2012-10-22'), open: 87.4885, high: 90.7685, low:
84.4285,close: 86.2857,volume: 866040680 },
    {x: new Date('2012-10-29'), open: 84.9828, high: 86.1428, low:
82.1071,close: 82.4,volume: 367371310},
    {x: new Date('2012-11-05'), open: 83.3593, high: 84.3914, low:
76.2457,close: 78.1514,volume: 919719846},
    {x: new Date('2012-11-12'), open: 79.1643, high: 79.2143, low:
72.25,close: 75.3825,volume: 894382149},
    {x: new Date('2012-11-19'), open: 77.2443, high: 81.7143, low:
77.1257,close: 81.6428,volume: 527416747},
    {x: new Date('2012-11-26'), open: 82.2714, high: 84.8928, low:
81.7514,close: 83.6114,volume: 646467974},
    {x: new Date('2012-12-03'), open: 84.8071, high: 84.9414, low:
74.09,close: 76.1785,volume: 980096264},
    {x: new Date('2012-12-10'), open: 75, high: 78.5085, low: 72.2257,close:
72.8277,volume: 835016110},
    {x: new Date('2012-12-17'), open: 72.7043, high: 76.4143, low:
71.6043,close: 74.19,volume: 726150329},

```

```

{x: new Date('2012-12-24'), open: 74.3357, high: 74.8928, low:
72.0943,close: 72.7984,volume: 321104733},
{x: new Date('2012-12-31'), open: 72.9328, high: 79.2857, low:
72.7143,close: 75.2857,volume: 540854882},
{x: new Date('2013-01-07'), open: 74.5714, high: 75.9843, low:
73.6,close: 74.3285,volume: 574594262},
{x: new Date('2013-01-14'), open: 71.8114, high: 72.9643, low:
69.0543,close: 71.4285,volume: 803105621},
{x: new Date('2013-01-21'), open: 72.08, high: 73.57, low:
62.1428,close: 62.84,volume: 971912560},
{x: new Date('2013-01-28'), open: 62.5464, high: 66.0857, low:
62.2657,close: 64.8028,volume: 656549587},
{x: new Date('2013-02-04'), open: 64.8443, high: 68.4014, low:
63.1428,close: 67.8543,volume: 743778993},
{x: new Date('2013-02-11'), open: 68.0714, high: 69.2771, low:
65.7028,close: 65.7371,volume: 585292366},
{x: new Date('2013-02-18'), open: 65.8714, high: 66.1043, low:
63.26,close: 64.4014,volume: 421766997},
{x: new Date('2013-02-25'), open: 64.8357, high: 65.0171, low:
61.4257,close: 61.4957,volume: 582741215},
{x: new Date('2013-03-04'), open: 61.1143, high: 62.2043, low:
59.8571,close: 61.6743,volume: 632856539},
{x: new Date('2013-03-11'), open: 61.3928, high: 63.4614, low:
60.7343,close: 63.38,volume: 572066981},
{x: new Date('2013-03-18'), open: 63.0643, high: 66.0143, low:
63.0286,close: 65.9871,volume: 552156035},
{x: new Date('2013-03-25'), open: 66.3843, high: 67.1357, low:
63.0886,close: 63.2371,volume: 390762517},
{x: new Date('2013-04-01'), open: 63.1286, high: 63.3854, low:
59.9543,close: 60.4571,volume: 505273732},
{x: new Date('2013-04-08'), open: 60.6928, high: 62.57, low:
60.3557,close: 61.4,volume: 387323550},
{x: new Date('2013-04-15'), open: 61, high: 61.1271, low: 55.0143,close:
55.79,volume: 709945604},
{x: new Date('2013-04-22'), open: 56.0914, high: 59.8241, low:
55.8964,close: 59.6007,volume: 787007506},
{x: new Date('2013-04-29'), open: 60.0643, high: 64.7471, low: 60,close:
64.2828,volume: 655020017},
{x: new Date('2013-05-06'), open: 65.1014, high: 66.5357, low:
64.3543,close: 64.71,volume: 545488533},
{x: new Date('2013-05-13'), open: 64.5014, high: 65.4143, low:
59.8428,close: 61.8943,volume: 633706550},
{x: new Date('2013-05-20'), open: 61.7014, high: 64.05, low:
61.4428,close: 63.5928,volume: 494379068},
{x: new Date('2013-05-27'), open: 64.2714, high: 65.3, low:
62.7714,close: 64.2478,volume: 362907830},
{x: new Date('2013-06-03'), open: 64.39, high: 64.9186, low:
61.8243,close: 63.1158,volume: 443249793},
{x: new Date('2013-06-10'), open: 63.5328, high: 64.1541, low:
61.2143,close: 61.4357,volume: 389680092},
{x: new Date('2013-06-17'), open: 61.6343, high: 62.2428, low:
58.3,close: 59.0714,volume: 400384818},
{x: new Date('2013-06-24'), open: 58.2, high: 58.38, low: 55.5528,close:
56.6471,volume: 519314826},
{x: new Date('2013-07-01'), open: 57.5271, high: 60.47, low:
57.3171,close: 59.6314,volume: 343878841},

```

```

    {x: new Date('2013-07-08'), open: 60.0157, high: 61.3986, low:
58.6257,close: 60.93,volume: 384106977},
    {x: new Date('2013-07-15'), open: 60.7157, high: 62.1243, low:
60.5957,close: 60.7071,volume: 286035513},
    {x: new Date('2013-07-22'), open: 61.3514, high: 63.5128, low:
59.8157,close: 62.9986,volume: 395816827},
    {x: new Date('2013-07-29'), open: 62.9714, high: 66.1214, low:
62.8857,close: 66.0771,volume: 339668858},
    {x: new Date('2013-08-12'), open: 65.2657, high: 72.0357, low:
65.2328,close: 71.7614,volume: 711563584},
    {x: new Date('2013-08-19'), open: 72.0485, high: 73.3914, low:
71.1714,close: 71.5743,volume: 417119660},
    {x: new Date('2013-08-26'), open: 71.5357, high: 72.8857, low:
69.4286,close: 69.6023,volume: 392805888},
    {x: new Date('2013-09-02'), open: 70.4428, high: 71.7485, low:
69.6214,close: 71.1743,volume: 317244380},
    {x: new Date('2013-09-09'), open: 72.1428, high: 72.56, low:
66.3857,close: 66.4143,volume: 669376320},
    {x: new Date('2013-09-16'), open: 65.8571, high: 68.3643, low:
63.8886,close: 66.7728,volume: 625142677},
    {x: new Date('2013-09-23'), open: 70.8714, high: 70.9871, low:
68.6743,close: 68.9643,volume: 475274537},
    {x: new Date('2013-09-30'), open: 68.1786, high: 70.3357, low:
67.773,close: 69.0043,volume: 368198906},
    {x: new Date('2013-10-07'), open: 69.5086, high: 70.5486, low:
68.3257,close: 70.4017,volume: 361437661},
    {x: new Date('2013-10-14'), open: 69.9757, high: 72.7514, low:
69.9071,close: 72.6985,volume: 342694379},
    {x: new Date('2013-10-21'), open: 73.11, high: 76.1757, low:
72.5757,close: 75.1368,volume: 490458997},
    {x: new Date('2013-10-28'), open: 75.5771, high: 77.0357, low:
73.5057,close: 74.29,volume: 508130174},
    {x: new Date('2013-11-04'), open: 74.4428, high: 75.555, low:
73.1971,close: 74.3657,volume: 318132218},
    {x: new Date('2013-11-11'), open: 74.2843, high: 75.6114, low:
73.4871,close: 74.9987,volume: 306711021},
    {x: new Date('2013-11-18'), open: 74.9985, high: 75.3128, low:
73.3814,close: 74.2571,volume: 282778778},
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        title: 'Months',
        valueType: 'DateTime',
        intervalType: 'Months',
        majorGridLines: { width: 0},
        crosshairTooltip: { enable: true },
    },
    primaryYAxis: {
        title: 'Price (Million Dollars)',
        minimum: 30,
        maximum: 180,
        interval: 30
    },
    series:[{
        dataSource: chartData, width: 2, low: 'low', high: 'high', close:
'close', open: 'open',
        xName: 'x', yName: 'y',

```

```

        name: 'USA',
        type: 'Candle',
    }],
    indicators: [{
        type: 'BollingerBands', field: 'Close', seriesName: 'USA', fill:
'blue',
        period: 3, animation: { enable: true }, upperLine: { color: 'orange'
}, lowerLine: { color: 'yellow' }
    }],
    tooltip: { enable: true, shared: true },
    chartArea: { border: { width: 0 } },
    crosshair: { enable: true, lineType: 'Vertical' },
    title: 'AAPL - 2016/2017'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Customization of BollingerBand

stroke, stroke-width, and color of upperLine can be customized by using [upperLine](#), and the lowerLine can be customized by using [lowerLine](#) properties of indicator.

INDEX.JS

```

var chartData = [
  {x: new Date('2012-10-15'), open: 90.3357, high: 93.2557, low:
87.0885, close: 87.12, volume: 646996264},

```

```

{x: new Date('2012-10-22'), open: 87.4885, high: 90.7685, low:
84.4285,close: 86.2857,volume: 866040680 },
{x: new Date('2012-10-29'), open: 84.9828, high: 86.1428, low:
82.1071,close: 82.4,volume: 367371310},
{x: new Date('2012-11-05'), open: 83.3593, high: 84.3914, low:
76.2457,close: 78.1514,volume: 919719846},
{x: new Date('2012-11-12'), open: 79.1643, high: 79.2143, low:
72.25,close: 75.3825,volume: 894382149},
{x: new Date('2012-11-19'), open: 77.2443, high: 81.7143, low:
77.1257,close: 81.6428,volume: 527416747},
{x: new Date('2012-11-26'), open: 82.2714, high: 84.8928, low:
81.7514,close: 83.6114,volume: 646467974},
{x: new Date('2012-12-03'), open: 84.8071, high: 84.9414, low:
74.09,close: 76.1785,volume: 980096264},
{x: new Date('2012-12-10'), open: 75, high: 78.5085, low: 72.2257,close:
72.8277,volume: 835016110},
{x: new Date('2012-12-17'), open: 72.7043, high: 76.4143, low:
71.6043,close: 74.19,volume: 726150329},
{x: new Date('2012-12-24'), open: 74.3357, high: 74.8928, low:
72.0943,close: 72.7984,volume: 321104733},
{x: new Date('2012-12-31'), open: 72.9328, high: 79.2857, low:
72.7143,close: 75.2857,volume: 540854882},
{x: new Date('2013-01-07'), open: 74.5714, high: 75.9843, low:
73.6,close: 74.3285,volume: 574594262},
{x: new Date('2013-01-14'), open: 71.8114, high: 72.9643, low:
69.0543,close: 71.4285,volume: 803105621},
{x: new Date('2013-01-21'), open: 72.08, high: 73.57, low:
62.1428,close: 62.84,volume: 971912560},
{x: new Date('2013-01-28'), open: 62.5464, high: 66.0857, low:
62.2657,close: 64.8028,volume: 656549587},
{x: new Date('2013-02-04'), open: 64.8443, high: 68.4014, low:
63.1428,close: 67.8543,volume: 743778993},
{x: new Date('2013-02-11'), open: 68.0714, high: 69.2771, low:
65.7028,close: 65.7371,volume: 585292366},
{x: new Date('2013-02-18'), open: 65.8714, high: 66.1043, low:
63.26,close: 64.4014,volume: 421766997},
{x: new Date('2013-02-25'), open: 64.8357, high: 65.0171, low:
61.4257,close: 61.4957,volume: 582741215},
{x: new Date('2013-03-04'), open: 61.1143, high: 62.2043, low:
59.8571,close: 61.6743,volume: 632856539},
{x: new Date('2013-03-11'), open: 61.3928, high: 63.4614, low:
60.7343,close: 63.38,volume: 572066981},
{x: new Date('2013-03-18'), open: 63.0643, high: 66.0143, low:
63.0286,close: 65.9871,volume: 552156035},
{x: new Date('2013-03-25'), open: 66.3843, high: 67.1357, low:
63.0886,close: 63.2371,volume: 390762517},
{x: new Date('2013-04-01'), open: 63.1286, high: 63.3854, low:
59.9543,close: 60.4571,volume: 505273732},
{x: new Date('2013-04-08'), open: 60.6928, high: 62.57, low:
60.3557,close: 61.4,volume: 387323550},
{x: new Date('2013-04-15'), open: 61, high: 61.1271, low: 55.0143,close:
55.79,volume: 709945604},
{x: new Date('2013-04-22'), open: 56.0914, high: 59.8241, low:
55.8964,close: 59.6007,volume: 787007506},
{x: new Date('2013-04-29'), open: 60.0643, high: 64.7471, low: 60,close:
64.2828,volume: 655020017},

```

```

    {x: new Date('2013-05-06'), open: 65.1014, high: 66.5357, low:
64.3543,close: 64.71,volume: 545488533},
    {x: new Date('2013-05-13'), open: 64.5014, high: 65.4143, low:
59.8428,close: 61.8943,volume: 633706550},
    {x: new Date('2013-05-20'), open: 61.7014, high: 64.05, low:
61.4428,close: 63.5928,volume: 494379068},
    {x: new Date('2013-05-27'), open: 64.2714, high: 65.3, low:
62.7714,close: 64.2478,volume: 362907830},
    {x: new Date('2013-06-03'), open: 64.39, high: 64.9186, low:
61.8243,close: 63.1158,volume: 443249793},
    {x: new Date('2013-06-10'), open: 63.5328, high: 64.1541, low:
61.2143,close: 61.4357,volume: 389680092},
    {x: new Date('2013-06-17'), open: 61.6343, high: 62.2428, low:
58.3,close: 59.0714,volume: 400384818},
    {x: new Date('2013-06-24'), open: 58.2, high: 58.38, low: 55.5528,close:
56.6471,volume: 519314826},
    {x: new Date('2013-07-01'), open: 57.5271, high: 60.47, low:
57.3171,close: 59.6314,volume: 343878841},
    {x: new Date('2013-07-08'), open: 60.0157, high: 61.3986, low:
58.6257,close: 60.93,volume: 384106977},
    {x: new Date('2013-07-15'), open: 60.7157, high: 62.1243, low:
60.5957,close: 60.7071,volume: 286035513},
    {x: new Date('2013-07-22'), open: 61.3514, high: 63.5128, low:
59.8157,close: 62.9986,volume: 395816827},
    {x: new Date('2013-07-29'), open: 62.9714, high: 66.1214, low:
62.8857,close: 66.0771,volume: 339668858},
    {x: new Date('2013-08-12'), open: 65.2657, high: 72.0357, low:
65.2328,close: 71.7614,volume: 711563584},
    {x: new Date('2013-08-19'), open: 72.0485, high: 73.3914, low:
71.1714,close: 71.5743,volume: 417119660},
    {x: new Date('2013-08-26'), open: 71.5357, high: 72.8857, low:
69.4286,close: 69.6023,volume: 392805888},
    {x: new Date('2013-09-02'), open: 70.4428, high: 71.7485, low:
69.6214,close: 71.1743,volume: 317244380},
    {x: new Date('2013-09-09'), open: 72.1428, high: 72.56, low:
66.3857,close: 66.4143,volume: 669376320},
    {x: new Date('2013-09-16'), open: 65.8571, high: 68.3643, low:
63.8886,close: 66.7728,volume: 625142677},
    {x: new Date('2013-09-23'), open: 70.8714, high: 70.9871, low:
68.6743,close: 68.9643,volume: 475274537},
    {x: new Date('2013-09-30'), open: 68.1786, high: 70.3357, low:
67.773,close: 69.0043,volume: 368198906},
    {x: new Date('2013-10-07'), open: 69.5086, high: 70.5486, low:
68.3257,close: 70.4017,volume: 361437661},
    {x: new Date('2013-10-14'), open: 69.9757, high: 72.7514, low:
69.9071,close: 72.6985,volume: 342694379},
    {x: new Date('2013-10-21'), open: 73.11, high: 76.1757, low:
72.5757,close: 75.1368,volume: 490458997},
    {x: new Date('2013-10-28'), open: 75.5771, high: 77.0357, low:
73.5057,close: 74.29,volume: 508130174},
    {x: new Date('2013-11-04'), open: 74.4428, high: 75.555, low:
73.1971,close: 74.3657,volume: 318132218},
    {x: new Date('2013-11-11'), open: 74.2843, high: 75.6114, low:
73.4871,close: 74.9987,volume: 306711021},
    {x: new Date('2013-11-18'), open: 74.9985, high: 75.3128, low:
73.3814,close: 74.2571,volume: 282778778},
];

```

```

let chart = new ej.charts.Chart({
  primaryXAxis: {
    title: 'Months',
    valueType: 'DateTime',
    intervalType: 'Months',
    majorGridLines: { width: 0 },
    crosshairTooltip: { enable: true },
  },
  primaryYAxis: {
    title: 'Price (Million Dollars)',
    minimum: 30,
    maximum: 180,
    interval: 30
  },
  series:[{
    dataSource: chartData, width: 2, low: 'low', high: 'high', close:
'close', open: 'open',
    xName: 'x', yName: 'y',
    name: 'USA',
    type: 'Candle',
  }],
  indicators: [{
    type: 'BollingerBands',
    field: 'Close', seriesName: 'USA',
    fill: 'blue',
    period: 3, upperLine: { color: 'red' },
    lowerLine: { color: 'green' }
  }],
  tooltip: { enable: true, shared: true },
  chartArea: { border: { width: 0 } },
  crosshair: { enable: true, lineType: 'Vertical' },
  title: 'AAPL - 2016/2017'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
</body>
</html>

```

```

var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Exponential Moving Average (EMA)

Moving average Indicators are used to define the direction of the trend. To render a EMA Indicator, use indicator [type](#) as `Ema` and inject `EmaIndicator` module using `Chart.Inject(EmaIndicator)`.

INDEX.JS

```

var chartData = [
    {x: new Date('2012-10-15'), open: 90.3357, high: 93.2557, low: 87.0885, close: 87.12, volume: 646996264},
    {x: new Date('2012-10-22'), open: 87.4885, high: 90.7685, low: 84.4285, close: 86.2857, volume: 866040680 },
    {x: new Date('2012-10-29'), open: 84.9828, high: 86.1428, low: 82.1071, close: 82.4, volume: 367371310},
    {x: new Date('2012-11-05'), open: 83.3593, high: 84.3914, low: 76.2457, close: 78.1514, volume: 919719846},
    {x: new Date('2012-11-12'), open: 79.1643, high: 79.2143, low: 72.25, close: 75.3825, volume: 894382149},
    {x: new Date('2012-11-19'), open: 77.2443, high: 81.7143, low: 77.1257, close: 81.6428, volume: 527416747},
    {x: new Date('2012-11-26'), open: 82.2714, high: 84.8928, low: 81.7514, close: 83.6114, volume: 646467974},
    {x: new Date('2012-12-03'), open: 84.8071, high: 84.9414, low: 74.09, close: 76.1785, volume: 980096264},
    {x: new Date('2012-12-10'), open: 75, high: 78.5085, low: 72.2257, close: 72.8277, volume: 835016110},
    {x: new Date('2012-12-17'), open: 72.7043, high: 76.4143, low: 71.6043, close: 74.19, volume: 726150329},
    {x: new Date('2012-12-24'), open: 74.3357, high: 74.8928, low: 72.0943, close: 72.7984, volume: 321104733},
    {x: new Date('2012-12-31'), open: 72.9328, high: 79.2857, low: 72.7143, close: 75.2857, volume: 540854882},
    {x: new Date('2013-01-07'), open: 74.5714, high: 75.9843, low: 73.6, close: 74.3285, volume: 574594262},
    {x: new Date('2013-01-14'), open: 71.8114, high: 72.9643, low: 69.0543, close: 71.4285, volume: 803105621},
    {x: new Date('2013-01-21'), open: 72.08, high: 73.57, low: 62.1428, close: 62.84, volume: 971912560},
    {x: new Date('2013-01-28'), open: 62.5464, high: 66.0857, low: 62.2657, close: 64.8028, volume: 656549587},
    {x: new Date('2013-02-04'), open: 64.8443, high: 68.4014, low: 63.1428, close: 67.8543, volume: 743778993},
    {x: new Date('2013-02-11'), open: 68.0714, high: 69.2771, low: 65.7028, close: 65.7371, volume: 585292366},
    {x: new Date('2013-02-18'), open: 65.8714, high: 66.1043, low: 63.26, close: 64.4014, volume: 421766997},
    {x: new Date('2013-02-25'), open: 64.8357, high: 65.0171, low: 61.4257, close: 61.4957, volume: 582741215},

```



```
{x: new Date('2013-03-04'), open: 61.1143, high: 62.2043, low: 59.8571, close: 61.6743, volume: 632856539},
{x: new Date('2013-03-11'), open: 61.3928, high: 63.4614, low: 60.7343, close: 63.38, volume: 572066981},
{x: new Date('2013-03-18'), open: 63.0643, high: 66.0143, low: 63.0286, close: 65.9871, volume: 552156035},
{x: new Date('2013-03-25'), open: 66.3843, high: 67.1357, low: 63.0886, close: 63.2371, volume: 390762517},
{x: new Date('2013-04-01'), open: 63.1286, high: 63.3854, low: 59.9543, close: 60.4571, volume: 505273732},
{x: new Date('2013-04-08'), open: 60.6928, high: 62.57, low: 60.3557, close: 61.4, volume: 387323550},
{x: new Date('2013-04-15'), open: 61, high: 61.1271, low: 55.0143, close: 55.79, volume: 709945604},
{x: new Date('2013-04-22'), open: 56.0914, high: 59.8241, low: 55.8964, close: 59.6007, volume: 787007506},
{x: new Date('2013-04-29'), open: 60.0643, high: 64.7471, low: 60, close: 64.2828, volume: 655020017},
{x: new Date('2013-05-06'), open: 65.1014, high: 66.5357, low: 64.3543, close: 64.71, volume: 545488533},
{x: new Date('2013-05-13'), open: 64.5014, high: 65.4143, low: 59.8428, close: 61.8943, volume: 633706550},
{x: new Date('2013-05-20'), open: 61.7014, high: 64.05, low: 61.4428, close: 63.5928, volume: 494379068},
{x: new Date('2013-05-27'), open: 64.2714, high: 65.3, low: 62.7714, close: 64.2478, volume: 362907830},
{x: new Date('2013-06-03'), open: 64.39, high: 64.9186, low: 61.8243, close: 63.1158, volume: 443249793},
{x: new Date('2013-06-10'), open: 63.5328, high: 64.1541, low: 61.2143, close: 61.4357, volume: 389680092},
{x: new Date('2013-06-17'), open: 61.6343, high: 62.2428, low: 58.3, close: 59.0714, volume: 400384818},
{x: new Date('2013-06-24'), open: 58.2, high: 58.38, low: 55.5528, close: 56.6471, volume: 519314826},
{x: new Date('2013-07-01'), open: 57.5271, high: 60.47, low: 57.3171, close: 59.6314, volume: 343878841},
{x: new Date('2013-07-08'), open: 60.0157, high: 61.3986, low: 58.6257, close: 60.93, volume: 384106977},
{x: new Date('2013-07-15'), open: 60.7157, high: 62.1243, low: 60.5957, close: 60.7071, volume: 286035513},
{x: new Date('2013-07-22'), open: 61.3514, high: 63.5128, low: 59.8157, close: 62.9986, volume: 395816827},
{x: new Date('2013-07-29'), open: 62.9714, high: 66.1214, low: 62.8857, close: 66.0771, volume: 339668858},
{x: new Date('2013-08-12'), open: 65.2657, high: 72.0357, low: 65.2328, close: 71.7614, volume: 711563584},
{x: new Date('2013-08-19'), open: 72.0485, high: 73.3914, low: 71.1714, close: 71.5743, volume: 417119660},
{x: new Date('2013-08-26'), open: 71.5357, high: 72.8857, low: 69.4286, close: 69.6023, volume: 392805888},
{x: new Date('2013-09-02'), open: 70.4428, high: 71.7485, low: 69.6214, close: 71.1743, volume: 317244380},
{x: new Date('2013-09-09'), open: 72.1428, high: 72.56, low: 66.3857, close: 66.4143, volume: 669376320},
{x: new Date('2013-09-16'), open: 65.8571, high: 68.3643, low: 63.8886, close: 66.7728, volume: 625142677},
```

```

    {x: new Date('2013-09-23'), open: 70.8714, high: 70.9871, low:
68.6743,close: 68.9643,volume: 475274537},
    {x: new Date('2013-09-30'), open: 68.1786, high: 70.3357, low:
67.773,close: 69.0043,volume: 368198906},
    {x: new Date('2013-10-07'), open: 69.5086, high: 70.5486, low:
68.3257,close: 70.4017,volume: 361437661},
    {x: new Date('2013-10-14'), open: 69.9757, high: 72.7514, low:
69.9071,close: 72.6985,volume: 342694379},
    {x: new Date('2013-10-21'), open: 73.11, high: 76.1757, low:
72.5757,close: 75.1368,volume: 490458997},
    {x: new Date('2013-10-28'), open: 75.5771, high: 77.0357, low:
73.5057,close: 74.29,volume: 508130174},
    {x: new Date('2013-11-04'), open: 74.4428, high: 75.555, low:
73.1971,close: 74.3657,volume: 318132218},
    {x: new Date('2013-11-11'), open: 74.2843, high: 75.6114, low:
73.4871,close: 74.9987,volume: 306711021},
    {x: new Date('2013-11-18'), open: 74.9985, high: 75.3128, low:
73.3814,close: 74.2571,volume: 282778778},
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    title: 'Months',
    valueType: 'DateTime',
    intervalType: 'Months',
    majorGridLines: { width: 0 },
    crosshairTooltip: { enable: true },
  },
  primaryYAxis: {
    title: 'Price (Million Dollars)',
    minimum: 30, maximum: 180, interval: 30,
    majorGridLines: { width: 0 }
  },
  axes: [{
    name: 'secondary',
    minimum: 30,
    maximum: 110,
    majorGridLines: { width: 0 },
    opposedPosition: true
  }],
  series:[{
    dataSource: chartData, width: 2,
    xName: 'x', yName: 'y', low: 'low', high: 'high', close: 'close',
    volume: 'volume', open: 'open',
    name: 'Apple Inc',
    //Series type as RangeColumn
    type: 'Candle', animation: { enable: true }
  }],
  indicators: [{
    type: 'Ema', field: 'Close', seriesName: 'Apple Inc', fill: 'blue',
    period: 3, animation: { enable: true }
  }],
  tooltip: { enable: true, shared: true },
  chartArea: { border: { width: 0 } },
  crosshair: { enable: true, lineType: 'Vertical' },
  title: 'AAPL - 2016/2017'
}, '#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Momentum

Momentum shows the speed at which the price of the stock is changing. To render a Momentum indicator, use indicator [type](#) as `Momentum` and inject `MomentumIndicator` module using `Chart.Inject(MomentumIndicator)` method. Momentum indicator will be represented by two lines (upperLine, signalLine). In momentum indicator the upperBand value is always renders at the value 100.

INDEX.JS

```

var chartData = [
  {x: new Date('2012-10-15'), open: 90.3357, high: 93.2557, low:
87.0885,close: 87.12,volume: 646996264},
  {x: new Date('2012-10-22'), open: 87.4885, high: 90.7685, low:
84.4285,close: 86.2857,volume: 866040680 },
  {x: new Date('2012-10-29'), open: 84.9828, high: 86.1428, low:
82.1071,close: 82.4,volume: 367371310},
  {x: new Date('2012-11-05'), open: 83.3593, high: 84.3914, low:
76.2457,close: 78.1514,volume: 919719846},
  {x: new Date('2012-11-12'), open: 79.1643, high: 79.2143, low:
72.25,close: 75.3825,volume: 894382149},
  {x: new Date('2012-11-19'), open: 77.2443, high: 81.7143, low:
77.1257,close: 81.6428,volume: 527416747},
  {x: new Date('2012-11-26'), open: 82.2714, high: 84.8928, low:
81.7514,close: 83.6114,volume: 646467974},

```

```

{x: new Date('2012-12-03'), open: 84.8071, high: 84.9414, low:
74.09,close: 76.1785,volume: 980096264},
{x: new Date('2012-12-10'), open: 75, high: 78.5085, low: 72.2257,close:
72.8277,volume: 835016110},
{x: new Date('2012-12-17'), open: 72.7043, high: 76.4143, low:
71.6043,close: 74.19,volume: 726150329},
{x: new Date('2012-12-24'), open: 74.3357, high: 74.8928, low:
72.0943,close: 72.7984,volume: 321104733},
{x: new Date('2012-12-31'), open: 72.9328, high: 79.2857, low:
72.7143,close: 75.2857,volume: 540854882},
{x: new Date('2013-01-07'), open: 74.5714, high: 75.9843, low:
73.6,close: 74.3285,volume: 574594262},
{x: new Date('2013-01-14'), open: 71.8114, high: 72.9643, low:
69.0543,close: 71.4285,volume: 803105621},
{x: new Date('2013-01-21'), open: 72.08, high: 73.57, low:
62.1428,close: 62.84,volume: 971912560},
{x: new Date('2013-01-28'), open: 62.5464, high: 66.0857, low:
62.2657,close: 64.8028,volume: 656549587},
{x: new Date('2013-02-04'), open: 64.8443, high: 68.4014, low:
63.1428,close: 67.8543,volume: 743778993},
{x: new Date('2013-02-11'), open: 68.0714, high: 69.2771, low:
65.7028,close: 65.7371,volume: 585292366},
{x: new Date('2013-02-18'), open: 65.8714, high: 66.1043, low:
63.26,close: 64.4014,volume: 421766997},
{x: new Date('2013-02-25'), open: 64.8357, high: 65.0171, low:
61.4257,close: 61.4957,volume: 582741215},
{x: new Date('2013-03-04'), open: 61.1143, high: 62.2043, low:
59.8571,close: 61.6743,volume: 632856539},
{x: new Date('2013-03-11'), open: 61.3928, high: 63.4614, low:
60.7343,close: 63.38,volume: 572066981},
{x: new Date('2013-03-18'), open: 63.0643, high: 66.0143, low:
63.0286,close: 65.9871,volume: 552156035},
{x: new Date('2013-03-25'), open: 66.3843, high: 67.1357, low:
63.0886,close: 63.2371,volume: 390762517},
{x: new Date('2013-04-01'), open: 63.1286, high: 63.3854, low:
59.9543,close: 60.4571,volume: 505273732},
{x: new Date('2013-04-08'), open: 60.6928, high: 62.57, low:
60.3557,close: 61.4,volume: 387323550},
{x: new Date('2013-04-15'), open: 61, high: 61.1271, low: 55.0143,close:
55.79,volume: 709945604},
{x: new Date('2013-04-22'), open: 56.0914, high: 59.8241, low:
55.8964,close: 59.6007,volume: 787007506},
{x: new Date('2013-04-29'), open: 60.0643, high: 64.7471, low: 60,close:
64.2828,volume: 655020017},
{x: new Date('2013-05-06'), open: 65.1014, high: 66.5357, low:
64.3543,close: 64.71,volume: 545488533},
{x: new Date('2013-05-13'), open: 64.5014, high: 65.4143, low:
59.8428,close: 61.8943,volume: 633706550},
{x: new Date('2013-05-20'), open: 61.7014, high: 64.05, low:
61.4428,close: 63.5928,volume: 494379068},
{x: new Date('2013-05-27'), open: 64.2714, high: 65.3, low:
62.7714,close: 64.2478,volume: 362907830},
{x: new Date('2013-06-03'), open: 64.39, high: 64.9186, low:
61.8243,close: 63.1158,volume: 443249793},
{x: new Date('2013-06-10'), open: 63.5328, high: 64.1541, low:
61.2143,close: 61.4357,volume: 389680092},

```

```

    {x: new Date('2013-06-17'), open: 61.6343, high: 62.2428, low:
58.3,close: 59.0714,volume: 400384818},
    {x: new Date('2013-06-24'), open: 58.2, high: 58.38, low: 55.5528,close:
56.6471,volume: 519314826},
    {x: new Date('2013-07-01'), open: 57.5271, high: 60.47, low:
57.3171,close: 59.6314,volume: 343878841},
    {x: new Date('2013-07-08'), open: 60.0157, high: 61.3986, low:
58.6257,close: 60.93,volume: 384106977},
    {x: new Date('2013-07-15'), open: 60.7157, high: 62.1243, low:
60.5957,close: 60.7071,volume: 286035513},
    {x: new Date('2013-07-22'), open: 61.3514, high: 63.5128, low:
59.8157,close: 62.9986,volume: 395816827},
    {x: new Date('2013-07-29'), open: 62.9714, high: 66.1214, low:
62.8857,close: 66.0771,volume: 339668858},
    {x: new Date('2013-08-12'), open: 65.2657, high: 72.0357, low:
65.2328,close: 71.7614,volume: 711563584},
    {x: new Date('2013-08-19'), open: 72.0485, high: 73.3914, low:
71.1714,close: 71.5743,volume: 417119660},
    {x: new Date('2013-08-26'), open: 71.5357, high: 72.8857, low:
69.4286,close: 69.6023,volume: 392805888},
    {x: new Date('2013-09-02'), open: 70.4428, high: 71.7485, low:
69.6214,close: 71.1743,volume: 317244380},
    {x: new Date('2013-09-09'), open: 72.1428, high: 72.56, low:
66.3857,close: 66.4143,volume: 669376320},
    {x: new Date('2013-09-16'), open: 65.8571, high: 68.3643, low:
63.8886,close: 66.7728,volume: 625142677},
    {x: new Date('2013-09-23'), open: 70.8714, high: 70.9871, low:
68.6743,close: 68.9643,volume: 475274537},
    {x: new Date('2013-09-30'), open: 68.1786, high: 70.3357, low:
67.773,close: 69.0043,volume: 368198906},
    {x: new Date('2013-10-07'), open: 69.5086, high: 70.5486, low:
68.3257,close: 70.4017,volume: 361437661},
    {x: new Date('2013-10-14'), open: 69.9757, high: 72.7514, low:
69.9071,close: 72.6985,volume: 342694379},
    {x: new Date('2013-10-21'), open: 73.11, high: 76.1757, low:
72.5757,close: 75.1368,volume: 490458997},
    {x: new Date('2013-10-28'), open: 75.5771, high: 77.0357, low:
73.5057,close: 74.29,volume: 508130174},
    {x: new Date('2013-11-04'), open: 74.4428, high: 75.555, low:
73.1971,close: 74.3657,volume: 318132218},
    {x: new Date('2013-11-11'), open: 74.2843, high: 75.6114, low:
73.4871,close: 74.9987,volume: 306711021},
    {x: new Date('2013-11-18'), open: 74.9985, high: 75.3128, low:
73.3814,close: 74.2571,volume: 282778778},
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        title: 'Months',
        valueType: 'DateTime',
        intervalType: 'Months',
        majorGridLines: { width: 0 },
        crosshairTooltip: { enable: true },
    },
    primaryYAxis: {
        title: 'Price (Million Dollars)',
        minimum: 30, maximum: 180, interval: 30,
        majorGridLines: { width: 0 }
    }
});

```

```

    },
    axes: [{
        name: 'secondary',
        minimum: 30,
        maximum: 110,
        majorGridLines: { width: 0 },
        opposedPosition: true
    }],
    series: [{
        dataSource: chartData, width: 2,
        xName: 'x', yName: 'y', low: 'low', high: 'high', close: 'close',
        open: 'open',
        name: 'USA', animation: { enable: true },
        // Series type as StepLine
        type: 'Candle',
    }],
    indicators: [{
        type: 'Momentum', field: 'Close', seriesName: 'USA', yAxisName:
        'secondary',
        upperLine: { color: 'red' },
        period: 3, animation: { enable: true }
    }],
    tooltip: { enable: true, shared: true },
    chartArea: { border: { width: 0 } },
    crosshair: { enable: true, lineType: 'Vertical' },
    title: 'AAPL - 2016/2017'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
<script src="index.js" type="text/javascript"></script>

```

```
</body></html>
```

Customization of MomentumIndicator

stroke, stroke-width, and color of upperLine can be customized by using [upperLine](#) property of indicator.

INDEX.JS

```
var chartData = [
  {x: new Date('2012-10-15'), open: 90.3357, high: 93.2557, low:
87.0885,close: 87.12,volume: 646996264},
  {x: new Date('2012-10-22'), open: 87.4885, high: 90.7685, low:
84.4285,close: 86.2857,volume: 866040680 },
  {x: new Date('2012-10-29'), open: 84.9828, high: 86.1428, low:
82.1071,close: 82.4,volume: 367371310},
  {x: new Date('2012-11-05'), open: 83.3593, high: 84.3914, low:
76.2457,close: 78.1514,volume: 919719846},
  {x: new Date('2012-11-12'), open: 79.1643, high: 79.2143, low:
72.25,close: 75.3825,volume: 894382149},
  {x: new Date('2012-11-19'), open: 77.2443, high: 81.7143, low:
77.1257,close: 81.6428,volume: 527416747},
  {x: new Date('2012-11-26'), open: 82.2714, high: 84.8928, low:
81.7514,close: 83.6114,volume: 646467974},
  {x: new Date('2012-12-03'), open: 84.8071, high: 84.9414, low:
74.09,close: 76.1785,volume: 980096264},
  {x: new Date('2012-12-10'), open: 75, high: 78.5085, low: 72.2257,close:
72.8277,volume: 835016110},
  {x: new Date('2012-12-17'), open: 72.7043, high: 76.4143, low:
71.6043,close: 74.19,volume: 726150329},
  {x: new Date('2012-12-24'), open: 74.3357, high: 74.8928, low:
72.0943,close: 72.7984,volume: 321104733},
  {x: new Date('2012-12-31'), open: 72.9328, high: 79.2857, low:
72.7143,close: 75.2857,volume: 540854882},
  {x: new Date('2013-01-07'), open: 74.5714, high: 75.9843, low:
73.6,close: 74.3285,volume: 574594262},
  {x: new Date('2013-01-14'), open: 71.8114, high: 72.9643, low:
69.0543,close: 71.4285,volume: 803105621},
  {x: new Date('2013-01-21'), open: 72.08, high: 73.57, low:
62.1428,close: 62.84,volume: 971912560},
  {x: new Date('2013-01-28'), open: 62.5464, high: 66.0857, low:
62.2657,close: 64.8028,volume: 656549587},
  {x: new Date('2013-02-04'), open: 64.8443, high: 68.4014, low:
63.1428,close: 67.8543,volume: 743778993},
  {x: new Date('2013-02-11'), open: 68.0714, high: 69.2771, low:
65.7028,close: 65.7371,volume: 585292366},
  {x: new Date('2013-02-18'), open: 65.8714, high: 66.1043, low:
63.26,close: 64.4014,volume: 421766997},
  {x: new Date('2013-02-25'), open: 64.8357, high: 65.0171, low:
61.4257,close: 61.4957,volume: 582741215},
  {x: new Date('2013-03-04'), open: 61.1143, high: 62.2043, low:
59.8571,close: 61.6743,volume: 632856539},
  {x: new Date('2013-03-11'), open: 61.3928, high: 63.4614, low:
60.7343,close: 63.38,volume: 572066981},
  {x: new Date('2013-03-18'), open: 63.0643, high: 66.0143, low:
63.0286,close: 65.9871,volume: 552156035},
```

```

{x: new Date('2013-03-25'), open: 66.3843, high: 67.1357, low:
63.0886,close: 63.2371,volume: 390762517},
{x: new Date('2013-04-01'), open: 63.1286, high: 63.3854, low:
59.9543,close: 60.4571,volume: 505273732},
{x: new Date('2013-04-08'), open: 60.6928, high: 62.57, low:
60.3557,close: 61.4,volume: 387323550},
{x: new Date('2013-04-15'), open: 61, high: 61.1271, low: 55.0143,close:
55.79,volume: 709945604},
{x: new Date('2013-04-22'), open: 56.0914, high: 59.8241, low:
55.8964,close: 59.6007,volume: 787007506},
{x: new Date('2013-04-29'), open: 60.0643, high: 64.7471, low: 60,close:
64.2828,volume: 655020017},
{x: new Date('2013-05-06'), open: 65.1014, high: 66.5357, low:
64.3543,close: 64.71,volume: 545488533},
{x: new Date('2013-05-13'), open: 64.5014, high: 65.4143, low:
59.8428,close: 61.8943,volume: 633706550},
{x: new Date('2013-05-20'), open: 61.7014, high: 64.05, low:
61.4428,close: 63.5928,volume: 494379068},
{x: new Date('2013-05-27'), open: 64.2714, high: 65.3, low:
62.7714,close: 64.2478,volume: 362907830},
{x: new Date('2013-06-03'), open: 64.39, high: 64.9186, low:
61.8243,close: 63.1158,volume: 443249793},
{x: new Date('2013-06-10'), open: 63.5328, high: 64.1541, low:
61.2143,close: 61.4357,volume: 389680092},
{x: new Date('2013-06-17'), open: 61.6343, high: 62.2428, low:
58.3,close: 59.0714,volume: 400384818},
{x: new Date('2013-06-24'), open: 58.2, high: 58.38, low: 55.5528,close:
56.6471,volume: 519314826},
{x: new Date('2013-07-01'), open: 57.5271, high: 60.47, low:
57.3171,close: 59.6314,volume: 343878841},
{x: new Date('2013-07-08'), open: 60.0157, high: 61.3986, low:
58.6257,close: 60.93,volume: 384106977},
{x: new Date('2013-07-15'), open: 60.7157, high: 62.1243, low:
60.5957,close: 60.7071,volume: 286035513},
{x: new Date('2013-07-22'), open: 61.3514, high: 63.5128, low:
59.8157,close: 62.9986,volume: 395816827},
{x: new Date('2013-07-29'), open: 62.9714, high: 66.1214, low:
62.8857,close: 66.0771,volume: 339668858},
{x: new Date('2013-08-12'), open: 65.2657, high: 72.0357, low:
65.2328,close: 71.7614,volume: 711563584},
{x: new Date('2013-08-19'), open: 72.0485, high: 73.3914, low:
71.1714,close: 71.5743,volume: 417119660},
{x: new Date('2013-08-26'), open: 71.5357, high: 72.8857, low:
69.4286,close: 69.6023,volume: 392805888},
{x: new Date('2013-09-02'), open: 70.4428, high: 71.7485, low:
69.6214,close: 71.1743,volume: 317244380},
{x: new Date('2013-09-09'), open: 72.1428, high: 72.56, low:
66.3857,close: 66.4143,volume: 669376320},
{x: new Date('2013-09-16'), open: 65.8571, high: 68.3643, low:
63.8886,close: 66.7728,volume: 625142677},
{x: new Date('2013-09-23'), open: 70.8714, high: 70.9871, low:
68.6743,close: 68.9643,volume: 475274537},
{x: new Date('2013-09-30'), open: 68.1786, high: 70.3357, low:
67.773,close: 69.0043,volume: 368198906},
{x: new Date('2013-10-07'), open: 69.5086, high: 70.5486, low:
68.3257,close: 70.4017,volume: 361437661},

```



```

    {x: new Date('2013-10-14'), open: 69.9757, high: 72.7514, low:
69.9071,close: 72.6985,volume: 342694379},
    {x: new Date('2013-10-21'), open: 73.11, high: 76.1757, low:
72.5757,close: 75.1368,volume: 490458997},
    {x: new Date('2013-10-28'), open: 75.5771, high: 77.0357, low:
73.5057,close: 74.29,volume: 508130174},
    {x: new Date('2013-11-04'), open: 74.4428, high: 75.555, low:
73.1971,close: 74.3657,volume: 318132218},
    {x: new Date('2013-11-11'), open: 74.2843, high: 75.6114, low:
73.4871,close: 74.9987,volume: 306711021},
    {x: new Date('2013-11-18'), open: 74.9985, high: 75.3128, low:
73.3814,close: 74.2571,volume: 282778778},
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        title: 'Months',
        valueType: 'DateTime',
        intervalType: 'Months',
        majorGridLines: { width: 0 },
        crosshairTooltip: { enable: true },
    },
    primaryYAxis: {
        title: 'Price (Million Dollars)',
        minimum: 30, maximum: 180, interval: 30
    },
    axes: [{
        name: 'secondary',
        minimum: 30,
        maximum: 110,
        majorGridLines: { width: 0 },
        opposedPosition: true
    }],
    series:[{
        dataSource: chartData, width: 2,
        xName: 'x', yName: 'y', low: 'low', high: 'high', close: 'close',
open: 'open',
        name: 'USA', animation: { enable: true },
        // Series type as StepLine
        type: 'Candle',
    }],
    indicators: [{
        type: 'Momentum', field: 'Close', seriesName: 'USA', yAxisName:
'secondary',
        upperLine: { color: 'red' },
        period: 3, animation: { enable: true },
        upperLine: { color: 'green' }
    }],
    tooltip: { enable: true, shared: true },
    chartArea: { border: { width: 0 } },
    crosshair: { enable: true, lineType: 'Vertical' },
    title: 'AAPL - 2016/2017'
}, '#element');

```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
```

```

<title>EJ2 Animation</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Moving Average Convergence Divergence (MACD)

MACD is based on the difference between two EMA's. To render a MACD Indicator, use indicator [type](#) as

Macd and inject **MacdIndicator** module using **Chart.Inject(MacdIndicator)**. MACD indicator will be represented by MACD line, signal line, MACD histogram. MACD histogram is used to differentiate MACD line and signal line.

INDEX.JS

```

var chartData = [
    {x: new Date('2012-10-15'), open: 90.3357, high: 93.2557, low:
87.0885, close: 87.12, volume: 646996264},
    {x: new Date('2012-10-22'), open: 87.4885, high: 90.7685, low:
84.4285, close: 86.2857, volume: 866040680 },
    {x: new Date('2012-10-29'), open: 84.9828, high: 86.1428, low:
82.1071, close: 82.4, volume: 367371310},
    {x: new Date('2012-11-05'), open: 83.3593, high: 84.3914, low:
76.2457, close: 78.1514, volume: 919719846},
    {x: new Date('2012-11-12'), open: 79.1643, high: 79.2143, low:
72.25, close: 75.3825, volume: 894382149},
    {x: new Date('2012-11-19'), open: 77.2443, high: 81.7143, low:
77.1257, close: 81.6428, volume: 527416747},
    {x: new Date('2012-11-26'), open: 82.2714, high: 84.8928, low:
81.7514, close: 83.6114, volume: 646467974},
    {x: new Date('2012-12-03'), open: 84.8071, high: 84.9414, low:
74.09, close: 76.1785, volume: 980096264},
    {x: new Date('2012-12-10'), open: 75, high: 78.5085, low: 72.2257, close:
72.8277, volume: 835016110},

```

```

{x: new Date('2012-12-17'), open: 72.7043, high: 76.4143, low:
71.6043,close: 74.19,volume: 726150329},
{x: new Date('2012-12-24'), open: 74.3357, high: 74.8928, low:
72.0943,close: 72.7984,volume: 321104733},
{x: new Date('2012-12-31'), open: 72.9328, high: 79.2857, low:
72.7143,close: 75.2857,volume: 540854882},
{x: new Date('2013-01-07'), open: 74.5714, high: 75.9843, low:
73.6,close: 74.3285,volume: 574594262},
{x: new Date('2013-01-14'), open: 71.8114, high: 72.9643, low:
69.0543,close: 71.4285,volume: 803105621},
{x: new Date('2013-01-21'), open: 72.08, high: 73.57, low:
62.1428,close: 62.84,volume: 971912560},
{x: new Date('2013-01-28'), open: 62.5464, high: 66.0857, low:
62.2657,close: 64.8028,volume: 656549587},
{x: new Date('2013-02-04'), open: 64.8443, high: 68.4014, low:
63.1428,close: 67.8543,volume: 743778993},
{x: new Date('2013-02-11'), open: 68.0714, high: 69.2771, low:
65.7028,close: 65.7371,volume: 585292366},
{x: new Date('2013-02-18'), open: 65.8714, high: 66.1043, low:
63.26,close: 64.4014,volume: 421766997},
{x: new Date('2013-02-25'), open: 64.8357, high: 65.0171, low:
61.4257,close: 61.4957,volume: 582741215},
{x: new Date('2013-03-04'), open: 61.1143, high: 62.2043, low:
59.8571,close: 61.6743,volume: 632856539},
{x: new Date('2013-03-11'), open: 61.3928, high: 63.4614, low:
60.7343,close: 63.38,volume: 572066981},
{x: new Date('2013-03-18'), open: 63.0643, high: 66.0143, low:
63.0286,close: 65.9871,volume: 552156035},
{x: new Date('2013-03-25'), open: 66.3843, high: 67.1357, low:
63.0886,close: 63.2371,volume: 390762517},
{x: new Date('2013-04-01'), open: 63.1286, high: 63.3854, low:
59.9543,close: 60.4571,volume: 505273732},
{x: new Date('2013-04-08'), open: 60.6928, high: 62.57, low:
60.3557,close: 61.4,volume: 387323550},
{x: new Date('2013-04-15'), open: 61, high: 61.1271, low: 55.0143,close:
55.79,volume: 709945604},
{x: new Date('2013-04-22'), open: 56.0914, high: 59.8241, low:
55.8964,close: 59.6007,volume: 787007506},
{x: new Date('2013-04-29'), open: 60.0643, high: 64.7471, low: 60,close:
64.2828,volume: 655020017},
{x: new Date('2013-05-06'), open: 65.1014, high: 66.5357, low:
64.3543,close: 64.71,volume: 545488533},
{x: new Date('2013-05-13'), open: 64.5014, high: 65.4143, low:
59.8428,close: 61.8943,volume: 633706550},
{x: new Date('2013-05-20'), open: 61.7014, high: 64.05, low:
61.4428,close: 63.5928,volume: 494379068},
{x: new Date('2013-05-27'), open: 64.2714, high: 65.3, low:
62.7714,close: 64.2478,volume: 362907830},
{x: new Date('2013-06-03'), open: 64.39, high: 64.9186, low:
61.8243,close: 63.1158,volume: 443249793},
{x: new Date('2013-06-10'), open: 63.5328, high: 64.1541, low:
61.2143,close: 61.4357,volume: 389680092},
{x: new Date('2013-06-17'), open: 61.6343, high: 62.2428, low:
58.3,close: 59.0714,volume: 400384818},
{x: new Date('2013-06-24'), open: 58.2, high: 58.38, low: 55.5528,close:
56.6471,volume: 519314826},

```

```

    {x: new Date('2013-07-01'), open: 57.5271, high: 60.47, low:
57.3171,close: 59.6314,volume: 343878841},
    {x: new Date('2013-07-08'), open: 60.0157, high: 61.3986, low:
58.6257,close: 60.93,volume: 384106977},
    {x: new Date('2013-07-15'), open: 60.7157, high: 62.1243, low:
60.5957,close: 60.7071,volume: 286035513},
    {x: new Date('2013-07-22'), open: 61.3514, high: 63.5128, low:
59.8157,close: 62.9986,volume: 395816827},
    {x: new Date('2013-07-29'), open: 62.9714, high: 66.1214, low:
62.8857,close: 66.0771,volume: 339668858},
    {x: new Date('2013-08-12'), open: 65.2657, high: 72.0357, low:
65.2328,close: 71.7614,volume: 711563584},
    {x: new Date('2013-08-19'), open: 72.0485, high: 73.3914, low:
71.1714,close: 71.5743,volume: 417119660},
    {x: new Date('2013-08-26'), open: 71.5357, high: 72.8857, low:
69.4286,close: 69.6023,volume: 392805888},
    {x: new Date('2013-09-02'), open: 70.4428, high: 71.7485, low:
69.6214,close: 71.1743,volume: 317244380},
    {x: new Date('2013-09-09'), open: 72.1428, high: 72.56, low:
66.3857,close: 66.4143,volume: 669376320},
    {x: new Date('2013-09-16'), open: 65.8571, high: 68.3643, low:
63.8886,close: 66.7728,volume: 625142677},
    {x: new Date('2013-09-23'), open: 70.8714, high: 70.9871, low:
68.6743,close: 68.9643,volume: 475274537},
    {x: new Date('2013-09-30'), open: 68.1786, high: 70.3357, low:
67.773,close: 69.0043,volume: 368198906},
    {x: new Date('2013-10-07'), open: 69.5086, high: 70.5486, low:
68.3257,close: 70.4017,volume: 361437661},
    {x: new Date('2013-10-14'), open: 69.9757, high: 72.7514, low:
69.9071,close: 72.6985,volume: 342694379},
    {x: new Date('2013-10-21'), open: 73.11, high: 76.1757, low:
72.5757,close: 75.1368,volume: 490458997},
    {x: new Date('2013-10-28'), open: 75.5771, high: 77.0357, low:
73.5057,close: 74.29,volume: 508130174},
    {x: new Date('2013-11-04'), open: 74.4428, high: 75.555, low:
73.1971,close: 74.3657,volume: 318132218},
    {x: new Date('2013-11-11'), open: 74.2843, high: 75.6114, low:
73.4871,close: 74.9987,volume: 306711021},
    {x: new Date('2013-11-18'), open: 74.9985, high: 75.3128, low:
73.3814,close: 74.2571,volume: 282778778},
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        title: 'Months',
        valueType: 'DateTime',
        intervalType: 'Months',
        majorGridLines: { width: 0 },
        crosshairTooltip: { enable: true },
    },
    primaryYAxis: {
        title: 'Price (Million Dollars)',
        minimum: 30, maximum: 180, interval: 30,
    },
    axes: [{
        name: 'secondary',
        majorGridLines: { width: 0 },
        opposedPosition: true
    }

```

```

    }],
    series:[{
        name: 'gold',
        type: 'Candle',
        xName: 'x',
        low: 'low',
        high: 'high',
        open: 'open',
        animation: { enable: true },
        close: 'close',
        dataSource: chartData
    }],
    indicators: [{
        type: 'Macd',
        period: 3,
        fastPeriod: 5,
        slowPeriod: 2,
        seriesName: 'gold',
        macdType: 'Both',
        width: 2,
        fill: 'blue',
        yAxisName: 'secondary',
    }],
    tooltip: { enable: true, shared: true },
    chartArea: { border: { width: 0 } },
    crosshair: { enable: true, lineType: 'Vertical' },
    title: 'AAPL - 2016/2017'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>

```

```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Customization of MACD

stroke, stroke-width, and color of macdLine can be customized by using [macdLine](#),

property of indicator. The positive and negative changes of histogram can be customized by [macdPositiveColor](#) and [macdNegativeColor](#) properties. The `[macdType]` is used to define the type of MACD indicator. To customize the MACD period using [slowPeriod](#) and [fastPeriod](#) properties.

By default `macdType` as 'Both'.

INDEX.JS

```
var chartData = [
  {x: new Date('2012-10-15'), open: 90.3357, high: 93.2557, low:
87.0885,close: 87.12,volume: 646996264},
  {x: new Date('2012-10-22'), open: 87.4885, high: 90.7685, low:
84.4285,close: 86.2857,volume: 866040680 },
  {x: new Date('2012-10-29'), open: 84.9828, high: 86.1428, low:
82.1071,close: 82.4,volume: 367371310},
  {x: new Date('2012-11-05'), open: 83.3593, high: 84.3914, low:
76.2457,close: 78.1514,volume: 919719846},
  {x: new Date('2012-11-12'), open: 79.1643, high: 79.2143, low:
72.25,close: 75.3825,volume: 894382149},
  {x: new Date('2012-11-19'), open: 77.2443, high: 81.7143, low:
77.1257,close: 81.6428,volume: 527416747},
  {x: new Date('2012-11-26'), open: 82.2714, high: 84.8928, low:
81.7514,close: 83.6114,volume: 646467974},
  {x: new Date('2012-12-03'), open: 84.8071, high: 84.9414, low:
74.09,close: 76.1785,volume: 980096264},
  {x: new Date('2012-12-10'), open: 75, high: 78.5085, low: 72.2257,close:
72.8277,volume: 835016110},
  {x: new Date('2012-12-17'), open: 72.7043, high: 76.4143, low:
71.6043,close: 74.19,volume: 726150329},
  {x: new Date('2012-12-24'), open: 74.3357, high: 74.8928, low:
72.0943,close: 72.7984,volume: 321104733},
  {x: new Date('2012-12-31'), open: 72.9328, high: 79.2857, low:
72.7143,close: 75.2857,volume: 540854882},
  {x: new Date('2013-01-07'), open: 74.5714, high: 75.9843, low:
73.6,close: 74.3285,volume: 574594262},
  {x: new Date('2013-01-14'), open: 71.8114, high: 72.9643, low:
69.0543,close: 71.4285,volume: 803105621},
  {x: new Date('2013-01-21'), open: 72.08, high: 73.57, low:
62.1428,close: 62.84,volume: 971912560},
  {x: new Date('2013-01-28'), open: 62.5464, high: 66.0857, low:
62.2657,close: 64.8028,volume: 656549587},
  {x: new Date('2013-02-04'), open: 64.8443, high: 68.4014, low:
63.1428,close: 67.8543,volume: 743778993},
  {x: new Date('2013-02-11'), open: 68.0714, high: 69.2771, low:
65.7028,close: 65.7371,volume: 585292366},
  {x: new Date('2013-02-18'), open: 65.8714, high: 66.1043, low:
63.26,close: 64.4014,volume: 421766997},
  {x: new Date('2013-02-25'), open: 64.8357, high: 65.0171, low:
61.4257,close: 61.4957,volume: 582741215},
```

```

{x: new Date('2013-03-04'), open: 61.1143, high: 62.2043, low:
59.8571,close: 61.6743,volume: 632856539},
{x: new Date('2013-03-11'), open: 61.3928, high: 63.4614, low:
60.7343,close: 63.38,volume: 572066981},
{x: new Date('2013-03-18'), open: 63.0643, high: 66.0143, low:
63.0286,close: 65.9871,volume: 552156035},
{x: new Date('2013-03-25'), open: 66.3843, high: 67.1357, low:
63.0886,close: 63.2371,volume: 390762517},
{x: new Date('2013-04-01'), open: 63.1286, high: 63.3854, low:
59.9543,close: 60.4571,volume: 505273732},
{x: new Date('2013-04-08'), open: 60.6928, high: 62.57, low:
60.3557,close: 61.4,volume: 387323550},
{x: new Date('2013-04-15'), open: 61, high: 61.1271, low: 55.0143,close:
55.79,volume: 709945604},
{x: new Date('2013-04-22'), open: 56.0914, high: 59.8241, low:
55.8964,close: 59.6007,volume: 787007506},
{x: new Date('2013-04-29'), open: 60.0643, high: 64.7471, low: 60,close:
64.2828,volume: 655020017},
{x: new Date('2013-05-06'), open: 65.1014, high: 66.5357, low:
64.3543,close: 64.71,volume: 545488533},
{x: new Date('2013-05-13'), open: 64.5014, high: 65.4143, low:
59.8428,close: 61.8943,volume: 633706550},
{x: new Date('2013-05-20'), open: 61.7014, high: 64.05, low:
61.4428,close: 63.5928,volume: 494379068},
{x: new Date('2013-05-27'), open: 64.2714, high: 65.3, low:
62.7714,close: 64.2478,volume: 362907830},
{x: new Date('2013-06-03'), open: 64.39, high: 64.9186, low:
61.8243,close: 63.1158,volume: 443249793},
{x: new Date('2013-06-10'), open: 63.5328, high: 64.1541, low:
61.2143,close: 61.4357,volume: 389680092},
{x: new Date('2013-06-17'), open: 61.6343, high: 62.2428, low:
58.3,close: 59.0714,volume: 400384818},
{x: new Date('2013-06-24'), open: 58.2, high: 58.38, low: 55.5528,close:
56.6471,volume: 519314826},
{x: new Date('2013-07-01'), open: 57.5271, high: 60.47, low:
57.3171,close: 59.6314,volume: 343878841},
{x: new Date('2013-07-08'), open: 60.0157, high: 61.3986, low:
58.6257,close: 60.93,volume: 384106977},
{x: new Date('2013-07-15'), open: 60.7157, high: 62.1243, low:
60.5957,close: 60.7071,volume: 286035513},
{x: new Date('2013-07-22'), open: 61.3514, high: 63.5128, low:
59.8157,close: 62.9986,volume: 395816827},
{x: new Date('2013-07-29'), open: 62.9714, high: 66.1214, low:
62.8857,close: 66.0771,volume: 339668858},
{x: new Date('2013-08-12'), open: 65.2657, high: 72.0357, low:
65.2328,close: 71.7614,volume: 711563584},
{x: new Date('2013-08-19'), open: 72.0485, high: 73.3914, low:
71.1714,close: 71.5743,volume: 417119660},
{x: new Date('2013-08-26'), open: 71.5357, high: 72.8857, low:
69.4286,close: 69.6023,volume: 392805888},
{x: new Date('2013-09-02'), open: 70.4428, high: 71.7485, low:
69.6214,close: 71.1743,volume: 317244380},
{x: new Date('2013-09-09'), open: 72.1428, high: 72.56, low:
66.3857,close: 66.4143,volume: 669376320},
{x: new Date('2013-09-16'), open: 65.8571, high: 68.3643, low:
63.8886,close: 66.7728,volume: 625142677},

```

```

    {x: new Date('2013-09-23'), open: 70.8714, high: 70.9871, low:
68.6743,close: 68.9643,volume: 475274537},
    {x: new Date('2013-09-30'), open: 68.1786, high: 70.3357, low:
67.773,close: 69.0043,volume: 368198906},
    {x: new Date('2013-10-07'), open: 69.5086, high: 70.5486, low:
68.3257,close: 70.4017,volume: 361437661},
    {x: new Date('2013-10-14'), open: 69.9757, high: 72.7514, low:
69.9071,close: 72.6985,volume: 342694379},
    {x: new Date('2013-10-21'), open: 73.11, high: 76.1757, low:
72.5757,close: 75.1368,volume: 490458997},
    {x: new Date('2013-10-28'), open: 75.5771, high: 77.0357, low:
73.5057,close: 74.29,volume: 508130174},
    {x: new Date('2013-11-04'), open: 74.4428, high: 75.555, low:
73.1971,close: 74.3657,volume: 318132218},
    {x: new Date('2013-11-11'), open: 74.2843, high: 75.6114, low:
73.4871,close: 74.9987,volume: 306711021},
    {x: new Date('2013-11-18'), open: 74.9985, high: 75.3128, low:
73.3814,close: 74.2571,volume: 282778778},
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    title: 'Months',
    valueType: 'DateTime',
    intervalType: 'Months',
    majorGridLines: { width: 0 },
    crosshairTooltip: { enable: true },
  },
  primaryYAxis: {
    title: 'Price (Million Dollars)',
    minimum: 30, maximum: 180, interval: 30
  },
  axes: [{
    name: 'secondary',
    majorGridLines: { width: 0 },
    opposedPosition: true
  }],
  series:[{
    name: 'gold',
    type: 'Candle',
    xName: 'x',
    low: 'low',
    high: 'high',
    open: 'open',
    animation: { enable: true },
    close: 'close',
    dataSource: chartData
  }],
  indicators: [{
    type: 'Macd',
    period: 3,
    fastPeriod: 5,
    slowPeriod: 2,
    seriesName: 'gold',
    macdType: 'Both',
    width: 2,
    fill: 'blue',
    yAxisName: 'secondary',
  }],
});

```



```

    }},
    tooltip: { enable: true, shared: true },
    chartArea: { border: { width: 0 } },
    crosshair: { enable: true, lineType: 'Vertical' },
    title: 'AAPL - 2016/2017'
  }, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Relative Strength Index (RSI)

RSI shows how strongly a stock is moving in its current direction. To render a RSI Indicator, use indicator [type](#) as `Rsi` and inject `RsiIndicator` module using `Chart.Inject(RsiIndicator)`. RSI indicator will be represented

by three lines (upperBand, lowerBand, signalLine). The upperBand and lowerBand values are customized by [overBought](#) and [overSold](#) properties of indicator and the signalLine is calculated by RSI formula.

INDEX.JS

```

var chartData = [
  {x: new Date('2012-10-15'), open: 90.3357, high: 93.2557, low:
87.0885, close: 87.12, volume: 646996264},
  {x: new Date('2012-10-22'), open: 87.4885, high: 90.7685, low:
84.4285, close: 86.2857, volume: 866040680 },
  {x: new Date('2012-10-29'), open: 84.9828, high: 86.1428, low:
82.1071, close: 82.4, volume: 367371310},

```

```

{x: new Date('2012-11-05'), open: 83.3593, high: 84.3914, low:
76.2457,close: 78.1514,volume: 919719846},
{x: new Date('2012-11-12'), open: 79.1643, high: 79.2143, low:
72.25,close: 75.3825,volume: 894382149},
{x: new Date('2012-11-19'), open: 77.2443, high: 81.7143, low:
77.1257,close: 81.6428,volume: 527416747},
{x: new Date('2012-11-26'), open: 82.2714, high: 84.8928, low:
81.7514,close: 83.6114,volume: 646467974},
{x: new Date('2012-12-03'), open: 84.8071, high: 84.9414, low:
74.09,close: 76.1785,volume: 980096264},
{x: new Date('2012-12-10'), open: 75, high: 78.5085, low: 72.2257,close:
72.8277,volume: 835016110},
{x: new Date('2012-12-17'), open: 72.7043, high: 76.4143, low:
71.6043,close: 74.19,volume: 726150329},
{x: new Date('2012-12-24'), open: 74.3357, high: 74.8928, low:
72.0943,close: 72.7984,volume: 321104733},
{x: new Date('2012-12-31'), open: 72.9328, high: 79.2857, low:
72.7143,close: 75.2857,volume: 540854882},
{x: new Date('2013-01-07'), open: 74.5714, high: 75.9843, low:
73.6,close: 74.3285,volume: 574594262},
{x: new Date('2013-01-14'), open: 71.8114, high: 72.9643, low:
69.0543,close: 71.4285,volume: 803105621},
{x: new Date('2013-01-21'), open: 72.08, high: 73.57, low:
62.1428,close: 62.84,volume: 971912560},
{x: new Date('2013-01-28'), open: 62.5464, high: 66.0857, low:
62.2657,close: 64.8028,volume: 656549587},
{x: new Date('2013-02-11'), open: 68.0714, high: 69.2771, low:
65.7028,close: 65.7371,volume: 585292366},
{x: new Date('2013-02-18'), open: 65.8714, high: 66.1043, low:
63.26,close: 64.4014,volume: 421766997},
{x: new Date('2013-02-25'), open: 64.8357, high: 65.0171, low:
61.4257,close: 61.4957,volume: 582741215},
{x: new Date('2013-03-04'), open: 61.1143, high: 62.2043, low:
59.8571,close: 61.6743,volume: 632856539},
{x: new Date('2013-03-11'), open: 61.3928, high: 63.4614, low:
60.7343,close: 63.38,volume: 572066981},
{x: new Date('2013-03-18'), open: 63.0643, high: 66.0143, low:
63.0286,close: 65.9871,volume: 552156035},
{x: new Date('2013-03-25'), open: 66.3843, high: 67.1357, low:
63.0886,close: 63.2371,volume: 390762517},
{x: new Date('2013-04-01'), open: 63.1286, high: 63.3854, low:
59.9543,close: 60.4571,volume: 505273732},
{x: new Date('2013-04-08'), open: 60.6928, high: 62.57, low:
60.3557,close: 61.4,volume: 387323550},
{x: new Date('2013-04-15'), open: 61, high: 61.1271, low: 55.0143,close:
55.79,volume: 709945604},
{x: new Date('2013-04-22'), open: 56.0914, high: 59.8241, low:
55.8964,close: 59.6007,volume: 787007506},
{x: new Date('2013-04-29'), open: 60.0643, high: 64.7471, low: 60,close:
64.2828,volume: 655020017},
{x: new Date('2013-05-06'), open: 65.1014, high: 66.5357, low:
64.3543,close: 64.71,volume: 545488533},
{x: new Date('2013-05-13'), open: 64.5014, high: 65.4143, low:
59.8428,close: 61.8943,volume: 633706550},
{x: new Date('2013-05-20'), open: 61.7014, high: 64.05, low:
61.4428,close: 63.5928,volume: 494379068},

```

```

    {x: new Date('2013-05-27'), open: 64.2714, high: 65.3, low:
62.7714,close: 64.2478,volume: 362907830},
    {x: new Date('2013-06-03'), open: 64.39, high: 64.9186, low:
61.8243,close: 63.1158,volume: 443249793},
    {x: new Date('2013-06-10'), open: 63.5328, high: 64.1541, low:
61.2143,close: 61.4357,volume: 389680092},
    {x: new Date('2013-06-17'), open: 61.6343, high: 62.2428, low:
58.3,close: 59.0714,volume: 400384818},
    {x: new Date('2013-06-24'), open: 58.2, high: 58.38, low: 55.5528,close:
56.6471,volume: 519314826},
    {x: new Date('2013-07-01'), open: 57.5271, high: 60.47, low:
57.3171,close: 59.6314,volume: 343878841},
    {x: new Date('2013-07-08'), open: 60.0157, high: 61.3986, low:
58.6257,close: 60.93,volume: 384106977},
    {x: new Date('2013-07-15'), open: 60.7157, high: 62.1243, low:
60.5957,close: 60.7071,volume: 286035513},
    {x: new Date('2013-07-22'), open: 61.3514, high: 63.5128, low:
59.8157,close: 62.9986,volume: 395816827},
    {x: new Date('2013-07-29'), open: 62.9714, high: 66.1214, low:
62.8857,close: 66.0771,volume: 339668858},
    {x: new Date('2013-08-12'), open: 65.2657, high: 72.0357, low:
65.2328,close: 71.7614,volume: 711563584},
    {x: new Date('2013-08-19'), open: 72.0485, high: 73.3914, low:
71.1714,close: 71.5743,volume: 417119660},
    {x: new Date('2013-08-26'), open: 71.5357, high: 72.8857, low:
69.4286,close: 69.6023,volume: 392805888},
    {x: new Date('2013-09-02'), open: 70.4428, high: 71.7485, low:
69.6214,close: 71.1743,volume: 317244380},
    {x: new Date('2013-09-09'), open: 72.1428, high: 72.56, low:
66.3857,close: 66.4143,volume: 669376320},
    {x: new Date('2013-09-16'), open: 65.8571, high: 68.3643, low:
63.8886,close: 66.7728,volume: 625142677},
    {x: new Date('2013-09-23'), open: 70.8714, high: 70.9871, low:
68.6743,close: 68.9643,volume: 475274537},
    {x: new Date('2013-09-30'), open: 68.1786, high: 70.3357, low:
67.773,close: 69.0043,volume: 368198906},
    {x: new Date('2013-10-07'), open: 69.5086, high: 70.5486, low:
68.3257,close: 70.4017,volume: 361437661},
    {x: new Date('2013-10-14'), open: 69.9757, high: 72.7514, low:
69.9071,close: 72.6985,volume: 342694379},
    {x: new Date('2013-10-21'), open: 73.11, high: 76.1757, low:
72.5757,close: 75.1368,volume: 490458997},
    {x: new Date('2013-10-28'), open: 75.5771, high: 77.0357, low:
73.5057,close: 74.29,volume: 508130174},
    {x: new Date('2013-11-04'), open: 74.4428, high: 75.555, low:
73.1971,close: 74.3657,volume: 318132218},
    {x: new Date('2013-11-11'), open: 74.2843, high: 75.6114, low:
73.4871,close: 74.9987,volume: 306711021},
    {x: new Date('2013-11-18'), open: 74.9985, high: 75.3128, low:
73.3814,close: 74.2571,volume: 282778778},
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        title: 'Months',
        valueType: 'DateTime',
        intervalType: 'Months',
        majorGridLines: { width: 0 },

```

```

        crosshairTooltip: { enable: true },
    },
    primaryYAxis: {
        title: 'Price (Million Dollars)',
        minimum: 30, maximum: 180, interval: 30
    },
    axes: [{
        name: 'secondary',
        minimum: 10,
        maximum: 110,
        majorGridLines: { width: 0 },
        opposedPosition: true
    }],
    series:[{
        dataSource: chartData, width: 2, low: 'low', high: 'high', close:
'close', open: 'open',
        xName: 'x', yName: 'y',
        name: 'USA',
        type: 'Candle',
    }],
    indicators: [{
        type: 'Rsi', field: 'Close', seriesName: 'USA', yAxisName:
'secondary', fill: 'blue',
        showZones: true, overBought: 70, overSold: 30,
        period: 3, animation: { enable: true }, upperLine: { color: 'red' },
        lowerLine: { color: 'green' }
    }],
    tooltip: { enable: true, shared: true },
    chartArea: { border: { width: 0 } },
    crosshair: { enable: true, lineType: 'Vertical' },
    title: 'AAPL - 2016/2017'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');

```

```

if(ele) {
    ele.style.visibility = "visible";
}

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Simple Moving Average (SMA)

Moving average Indicators are used to define the direction of the trend. To render a SMA Indicator, use indicator [type](#) as `Sma` and inject `SmaIndicator` module using `Chart.Inject(SmaIndicator)`.

INDEX.JS

```

var chartData = [
    {x: new Date('2012-10-15'), open: 90.3357, high: 93.2557, low: 87.0885, close: 87.12, volume: 646996264},
    {x: new Date('2012-10-22'), open: 87.4885, high: 90.7685, low: 84.4285, close: 86.2857, volume: 866040680},
    {x: new Date('2012-10-29'), open: 84.9828, high: 86.1428, low: 82.1071, close: 82.4, volume: 367371310},
    {x: new Date('2012-11-05'), open: 83.3593, high: 84.3914, low: 76.2457, close: 78.1514, volume: 919719846},
    {x: new Date('2012-11-12'), open: 79.1643, high: 79.2143, low: 72.25, close: 75.3825, volume: 894382149},
    {x: new Date('2012-11-19'), open: 77.2443, high: 81.7143, low: 77.1257, close: 81.6428, volume: 527416747},
    {x: new Date('2012-11-26'), open: 82.2714, high: 84.8928, low: 81.7514, close: 83.6114, volume: 646467974},
    {x: new Date('2012-12-03'), open: 84.8071, high: 84.9414, low: 74.09, close: 76.1785, volume: 980096264},
    {x: new Date('2012-12-10'), open: 75, high: 78.5085, low: 72.2257, close: 72.8277, volume: 835016110},
    {x: new Date('2012-12-17'), open: 72.7043, high: 76.4143, low: 71.6043, close: 74.19, volume: 726150329},
    {x: new Date('2012-12-24'), open: 74.3357, high: 74.8928, low: 72.0943, close: 72.7984, volume: 321104733},
    {x: new Date('2012-12-31'), open: 72.9328, high: 79.2857, low: 72.7143, close: 75.2857, volume: 540854882},
    {x: new Date('2013-01-07'), open: 74.5714, high: 75.9843, low: 73.6, close: 74.3285, volume: 574594262},
    {x: new Date('2013-01-14'), open: 71.8114, high: 72.9643, low: 69.0543, close: 71.4285, volume: 803105621},
    {x: new Date('2013-01-21'), open: 72.08, high: 73.57, low: 62.1428, close: 62.84, volume: 971912560},
    {x: new Date('2013-01-28'), open: 62.5464, high: 66.0857, low: 62.2657, close: 64.8028, volume: 656549587},
    {x: new Date('2013-02-04'), open: 64.8443, high: 68.4014, low: 63.1428, close: 67.8543, volume: 743778993},
    {x: new Date('2013-02-11'), open: 68.0714, high: 69.2771, low: 65.7028, close: 65.7371, volume: 585292366},
    {x: new Date('2013-02-18'), open: 65.8714, high: 66.1043, low: 63.26, close: 64.4014, volume: 421766997},
    {x: new Date('2013-02-25'), open: 64.8357, high: 65.0171, low: 61.4257, close: 61.4957, volume: 582741215},
    {x: new Date('2013-03-04'), open: 61.1143, high: 62.2043, low: 59.8571, close: 61.6743, volume: 632856539},

```

```

{x: new Date('2013-03-11'), open: 61.3928, high: 63.4614, low:
60.7343,close: 63.38,volume: 572066981},
{x: new Date('2013-03-18'), open: 63.0643, high: 66.0143, low:
63.0286,close: 65.9871,volume: 552156035},
{x: new Date('2013-03-25'), open: 66.3843, high: 67.1357, low:
63.0886,close: 63.2371,volume: 390762517},
{x: new Date('2013-04-01'), open: 63.1286, high: 63.3854, low:
59.9543,close: 60.4571,volume: 505273732},
{x: new Date('2013-04-08'), open: 60.6928, high: 62.57, low:
60.3557,close: 61.4,volume: 387323550},
{x: new Date('2013-04-15'), open: 61, high: 61.1271, low: 55.0143,close:
55.79,volume: 709945604},
{x: new Date('2013-04-22'), open: 56.0914, high: 59.8241, low:
55.8964,close: 59.6007,volume: 787007506},
{x: new Date('2013-04-29'), open: 60.0643, high: 64.7471, low: 60,close:
64.2828,volume: 655020017},
{x: new Date('2013-05-06'), open: 65.1014, high: 66.5357, low:
64.3543,close: 64.71,volume: 545488533},
{x: new Date('2013-05-13'), open: 64.5014, high: 65.4143, low:
59.8428,close: 61.8943,volume: 633706550},
{x: new Date('2013-05-20'), open: 61.7014, high: 64.05, low:
61.4428,close: 63.5928,volume: 494379068},
{x: new Date('2013-05-27'), open: 64.2714, high: 65.3, low:
62.7714,close: 64.2478,volume: 362907830},
{x: new Date('2013-06-03'), open: 64.39, high: 64.9186, low:
61.8243,close: 63.1158,volume: 443249793},
{x: new Date('2013-06-10'), open: 63.5328, high: 64.1541, low:
61.2143,close: 61.4357,volume: 389680092},
{x: new Date('2013-06-17'), open: 61.6343, high: 62.2428, low:
58.3,close: 59.0714,volume: 400384818},
{x: new Date('2013-06-24'), open: 58.2, high: 58.38, low: 55.5528,close:
56.6471,volume: 519314826},
{x: new Date('2013-07-01'), open: 57.5271, high: 60.47, low:
57.3171,close: 59.6314,volume: 343878841},
{x: new Date('2013-07-08'), open: 60.0157, high: 61.3986, low:
58.6257,close: 60.93,volume: 384106977},
{x: new Date('2013-07-15'), open: 60.7157, high: 62.1243, low:
60.5957,close: 60.7071,volume: 286035513},
{x: new Date('2013-07-22'), open: 61.3514, high: 63.5128, low:
59.8157,close: 62.9986,volume: 395816827},
{x: new Date('2013-07-29'), open: 62.9714, high: 66.1214, low:
62.8857,close: 66.0771,volume: 339668858},
{x: new Date('2013-08-12'), open: 65.2657, high: 72.0357, low:
65.2328,close: 71.7614,volume: 711563584},
{x: new Date('2013-08-19'), open: 72.0485, high: 73.3914, low:
71.1714,close: 71.5743,volume: 417119660},
{x: new Date('2013-08-26'), open: 71.5357, high: 72.8857, low:
69.4286,close: 69.6023,volume: 392805888},
{x: new Date('2013-09-02'), open: 70.4428, high: 71.7485, low:
69.6214,close: 71.1743,volume: 317244380},
{x: new Date('2013-09-09'), open: 72.1428, high: 72.56, low:
66.3857,close: 66.4143,volume: 669376320},
{x: new Date('2013-09-16'), open: 65.8571, high: 68.3643, low:
63.8886,close: 66.7728,volume: 625142677},
{x: new Date('2013-09-23'), open: 70.8714, high: 70.9871, low:
68.6743,close: 68.9643,volume: 475274537},

```

```

    {x: new Date('2013-09-30'), open: 68.1786, high: 70.3357, low:
67.773,close: 69.0043,volume: 368198906},
    {x: new Date('2013-10-07'), open: 69.5086, high: 70.5486, low:
68.3257,close: 70.4017,volume: 361437661},
    {x: new Date('2013-10-14'), open: 69.9757, high: 72.7514, low:
69.9071,close: 72.6985,volume: 342694379},
    {x: new Date('2013-10-21'), open: 73.11, high: 76.1757, low:
72.5757,close: 75.1368,volume: 490458997},
    {x: new Date('2013-10-28'), open: 75.5771, high: 77.0357, low:
73.5057,close: 74.29,volume: 508130174},
    {x: new Date('2013-11-04'), open: 74.4428, high: 75.555, low:
73.1971,close: 74.3657,volume: 318132218},
    {x: new Date('2013-11-11'), open: 74.2843, high: 75.6114, low:
73.4871,close: 74.9987,volume: 306711021},
    {x: new Date('2013-11-18'), open: 74.9985, high: 75.3128, low:
73.3814,close: 74.2571,volume: 282778778},
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        title: 'Months',
        valueType: 'DateTime',
        intervalType: 'Months',
        majorGridLines: { width: 0 },
        crosshairTooltip: { enable: true },
    },
    primaryYAxis: {
        title: 'Price (Million Dollars)',
        minimum: 30, maximum: 180, interval: 30
    },
    axes: [{
        name: 'secondary',
        minimum: 30,
        maximum: 110,
        majorGridLines: { width: 0 },
        opposedPosition: true
    }],
    series:[{
        dataSource: chartData, width: 2,
        xName: 'x', yName: 'y', low: 'low', high: 'high', close: 'close',
        volume: 'volume', open: 'open',
        name: 'Apple Inc',
        //Series type as RangeColumn
        type: 'Candle', animation: { enable: true }
    }],
    indicators: [{
        type: 'Sma', field: 'Close', seriesName: 'Apple Inc', fill: 'blue',
        period: 3, animation: { enable: true }
    }],
    tooltip: { enable: true, shared: true },
    chartArea: { border: { width: 0 } },
    crosshair: { enable: true, lineType: 'Vertical' },
    title: 'AAPL - 2016/2017'
}, '#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Stochastic

It shows how a stock is, when compared to previous state. To render a Stochastic indicator, use indicator [type](#) as **Stochastic** and inject **StochasticIndicator** module using **Chart.Inject(StochasticIndicator)** method.

stochastic indicator will be represented by four lines (upperLine, lowerLine, periodLine, signalLine). In stochastic indicator the upperBand value and lowerBand value is customized by [overBought](#) and [overSold](#) properties of indicators and the periodLine and signalLine is render based on stochastic formula.

INDEX.JS

```

var chartData = [
  {x: new Date('2012-10-15'), open: 90.3357, high: 93.2557, low:
87.0885,close: 87.12,volume: 646996264},
  {x: new Date('2012-10-22'), open: 87.4885, high: 90.7685, low:
84.4285,close: 86.2857,volume: 866040680 },
  {x: new Date('2012-10-29'), open: 84.9828, high: 86.1428, low:
82.1071,close: 82.4,volume: 367371310},
  {x: new Date('2012-11-05'), open: 83.3593, high: 84.3914, low:
76.2457,close: 78.1514,volume: 919719846},
  {x: new Date('2012-11-12'), open: 79.1643, high: 79.2143, low:
72.25,close: 75.3825,volume: 894382149},
  {x: new Date('2012-11-19'), open: 77.2443, high: 81.7143, low:
77.1257,close: 81.6428,volume: 527416747},
  {x: new Date('2012-11-26'), open: 82.2714, high: 84.8928, low:
81.7514,close: 83.6114,volume: 646467974},

```



```

{x: new Date('2012-12-03'), open: 84.8071, high: 84.9414, low:
74.09,close: 76.1785,volume: 980096264},
{x: new Date('2012-12-10'), open: 75, high: 78.5085, low: 72.2257,close:
72.8277,volume: 835016110},
{x: new Date('2012-12-17'), open: 72.7043, high: 76.4143, low:
71.6043,close: 74.19,volume: 726150329},
{x: new Date('2012-12-24'), open: 74.3357, high: 74.8928, low:
72.0943,close: 72.7984,volume: 321104733},
{x: new Date('2012-12-31'), open: 72.9328, high: 79.2857, low:
72.7143,close: 75.2857,volume: 540854882},
{x: new Date('2013-01-07'), open: 74.5714, high: 75.9843, low:
73.6,close: 74.3285,volume: 574594262},
{x: new Date('2013-01-14'), open: 71.8114, high: 72.9643, low:
69.0543,close: 71.4285,volume: 803105621},
{x: new Date('2013-01-21'), open: 72.08, high: 73.57, low:
62.1428,close: 62.84,volume: 971912560},
{x: new Date('2013-01-28'), open: 62.5464, high: 66.0857, low:
62.2657,close: 64.8028,volume: 656549587},
{x: new Date('2013-02-04'), open: 64.8443, high: 68.4014, low:
63.1428,close: 67.8543,volume: 743778993},
{x: new Date('2013-02-11'), open: 68.0714, high: 69.2771, low:
65.7028,close: 65.7371,volume: 585292366},
{x: new Date('2013-02-18'), open: 65.8714, high: 66.1043, low:
63.26,close: 64.4014,volume: 421766997},
{x: new Date('2013-02-25'), open: 64.8357, high: 65.0171, low:
61.4257,close: 61.4957,volume: 582741215},
{x: new Date('2013-03-04'), open: 61.1143, high: 62.2043, low:
59.8571,close: 61.6743,volume: 632856539},
{x: new Date('2013-03-11'), open: 61.3928, high: 63.4614, low:
60.7343,close: 63.38,volume: 572066981},
{x: new Date('2013-03-18'), open: 63.0643, high: 66.0143, low:
63.0286,close: 65.9871,volume: 552156035},
{x: new Date('2013-03-25'), open: 66.3843, high: 67.1357, low:
63.0886,close: 63.2371,volume: 390762517},
{x: new Date('2013-04-01'), open: 63.1286, high: 63.3854, low:
59.9543,close: 60.4571,volume: 505273732},
{x: new Date('2013-04-08'), open: 60.6928, high: 62.57, low:
60.3557,close: 61.4,volume: 387323550},
{x: new Date('2013-04-15'), open: 61, high: 61.1271, low: 55.0143,close:
55.79,volume: 709945604},
{x: new Date('2013-04-22'), open: 56.0914, high: 59.8241, low:
55.8964,close: 59.6007,volume: 787007506},
{x: new Date('2013-04-29'), open: 60.0643, high: 64.7471, low: 60,close:
64.2828,volume: 655020017},
{x: new Date('2013-05-06'), open: 65.1014, high: 66.5357, low:
64.3543,close: 64.71,volume: 545488533},
{x: new Date('2013-05-13'), open: 64.5014, high: 65.4143, low:
59.8428,close: 61.8943,volume: 633706550},
{x: new Date('2013-05-20'), open: 61.7014, high: 64.05, low:
61.4428,close: 63.5928,volume: 494379068},
{x: new Date('2013-05-27'), open: 64.2714, high: 65.3, low:
62.7714,close: 64.2478,volume: 362907830},
{x: new Date('2013-06-03'), open: 64.39, high: 64.9186, low:
61.8243,close: 63.1158,volume: 443249793},
{x: new Date('2013-06-10'), open: 63.5328, high: 64.1541, low:
61.2143,close: 61.4357,volume: 389680092},

```

```

    {x: new Date('2013-06-17'), open: 61.6343, high: 62.2428, low:
58.3,close: 59.0714,volume: 400384818},
    {x: new Date('2013-06-24'), open: 58.2, high: 58.38, low: 55.5528,close:
56.6471,volume: 519314826},
    {x: new Date('2013-07-01'), open: 57.5271, high: 60.47, low:
57.3171,close: 59.6314,volume: 343878841},
    {x: new Date('2013-07-08'), open: 60.0157, high: 61.3986, low:
58.6257,close: 60.93,volume: 384106977},
    {x: new Date('2013-07-15'), open: 60.7157, high: 62.1243, low:
60.5957,close: 60.7071,volume: 286035513},
    {x: new Date('2013-07-22'), open: 61.3514, high: 63.5128, low:
59.8157,close: 62.9986,volume: 395816827},
    {x: new Date('2013-07-29'), open: 62.9714, high: 66.1214, low:
62.8857,close: 66.0771,volume: 339668858},
    {x: new Date('2013-08-12'), open: 65.2657, high: 72.0357, low:
65.2328,close: 71.7614,volume: 711563584},
    {x: new Date('2013-08-19'), open: 72.0485, high: 73.3914, low:
71.1714,close: 71.5743,volume: 417119660},
    {x: new Date('2013-08-26'), open: 71.5357, high: 72.8857, low:
69.4286,close: 69.6023,volume: 392805888},
    {x: new Date('2013-09-02'), open: 70.4428, high: 71.7485, low:
69.6214,close: 71.1743,volume: 317244380},
    {x: new Date('2013-09-09'), open: 72.1428, high: 72.56, low:
66.3857,close: 66.4143,volume: 669376320},
    {x: new Date('2013-09-16'), open: 65.8571, high: 68.3643, low:
63.8886,close: 66.7728,volume: 625142677},
    {x: new Date('2013-09-23'), open: 70.8714, high: 70.9871, low:
68.6743,close: 68.9643,volume: 475274537},
    {x: new Date('2013-09-30'), open: 68.1786, high: 70.3357, low:
67.773,close: 69.0043,volume: 368198906},
    {x: new Date('2013-10-07'), open: 69.5086, high: 70.5486, low:
68.3257,close: 70.4017,volume: 361437661},
    {x: new Date('2013-10-14'), open: 69.9757, high: 72.7514, low:
69.9071,close: 72.6985,volume: 342694379},
    {x: new Date('2013-10-21'), open: 73.11, high: 76.1757, low:
72.5757,close: 75.1368,volume: 490458997},
    {x: new Date('2013-10-28'), open: 75.5771, high: 77.0357, low:
73.5057,close: 74.29,volume: 508130174},
    {x: new Date('2013-11-04'), open: 74.4428, high: 75.555, low:
73.1971,close: 74.3657,volume: 318132218},
    {x: new Date('2013-11-11'), open: 74.2843, high: 75.6114, low:
73.4871,close: 74.9987,volume: 306711021},
    {x: new Date('2013-11-18'), open: 74.9985, high: 75.3128, low:
73.3814,close: 74.2571,volume: 282778778},
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        title: 'Months',
        valueType: 'DateTime',
        intervalType: 'Months',
        majorGridLines: { width: 0 },
        crosshairTooltip: { enable: true },
    },
    primaryYAxis: {
        title: 'Price (Million Dollars)',
        minimum: 30, maximum: 180, interval: 30
    },
},

```

```

axes: [{
    name: 'secondary',
    minimum: 10,
    maximum: 110,
    majorGridLines: { width: 0 },
    opposedPosition: true
}],
series:[{
    dataSource: chartData, width: 2, low: 'low', high: 'high', close:
'close', open: 'open',
    xName: 'x', yName: 'y',
    name: 'USA',
    type: 'Candle',
}],
indicators: [{
    type: 'Stochastic', field: 'Close', seriesName: 'USA', yAxisName:
'secondary', fill: 'blue',
    kPeriod: 2, dPeriod: 3, showZones: true,
    period: 3, animation: { enable: false },
}],
tooltip: { enable: true, shared: true },
chartArea: { border: { width: 0 } },
crosshair: { enable: true, lineType: 'Vertical' },
title: 'AAPL - 2016/2017'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Customization of StochasticIndicator

stroke, stroke-width, and color of upperLine can be customized by using [upperLine](#),

the lowerLine can be customized by using [lowerLine](#) and the periodLine can be customized by using [periodLine](#) properties of indicator. To customize the period to find the average price using [kPeriod](#) and [dPeriod](#)

properties.

INDEX.JS

```
var chartData = [
    {x: new Date('2012-10-15'), open: 90.3357, high: 93.2557, low:
87.0885,close: 87.12,volume: 646996264},
    {x: new Date('2012-10-22'), open: 87.4885, high: 90.7685, low:
84.4285,close: 86.2857,volume: 866040680 },
    {x: new Date('2012-10-29'), open: 84.9828, high: 86.1428, low:
82.1071,close: 82.4,volume: 367371310},
    {x: new Date('2012-11-05'), open: 83.3593, high: 84.3914, low:
76.2457,close: 78.1514,volume: 919719846},
    {x: new Date('2012-11-12'), open: 79.1643, high: 79.2143, low:
72.25,close: 75.3825,volume: 894382149},
    {x: new Date('2012-11-19'), open: 77.2443, high: 81.7143, low:
77.1257,close: 81.6428,volume: 527416747},
    {x: new Date('2012-11-26'), open: 82.2714, high: 84.8928, low:
81.7514,close: 83.6114,volume: 646467974},
    {x: new Date('2012-12-03'), open: 84.8071, high: 84.9414, low:
74.09,close: 76.1785,volume: 980096264},
    {x: new Date('2012-12-10'), open: 75, high: 78.5085, low: 72.2257,close:
72.8277,volume: 835016110},
    {x: new Date('2012-12-17'), open: 72.7043, high: 76.4143, low:
71.6043,close: 74.19,volume: 726150329},
    {x: new Date('2012-12-24'), open: 74.3357, high: 74.8928, low:
72.0943,close: 72.7984,volume: 321104733},
    {x: new Date('2012-12-31'), open: 72.9328, high: 79.2857, low: 72.7143,
close: 75.2857, volume: 540854882},
    {x: new Date('2013-01-07'), open: 74.5714, high: 75.9843, low:
73.6,close: 74.3285,volume: 574594262},
    {x: new Date('2013-01-14'), open: 71.8114, high: 72.9643, low:
69.0543,close: 71.4285,volume: 803105621},
    {x: new Date('2013-01-21'), open: 72.08, high: 73.57, low:
62.1428,close: 62.84,volume: 971912560},
    {x: new Date('2013-01-28'), open: 62.5464, high: 66.0857, low:
62.2657,close: 64.8028,volume: 656549587},
    {x: new Date('2013-02-04'), open: 64.8443, high: 68.4014, low:
63.1428,close: 67.8543,volume: 743778993},
    {x: new Date('2013-02-11'), open: 68.0714, high: 69.2771, low:
65.7028,close: 65.7371,volume: 585292366},
    {x: new Date('2013-02-18'), open: 65.8714, high: 66.1043, low:
63.26,close: 64.4014,volume: 421766997},
    {x: new Date('2013-02-25'), open: 64.8357, high: 65.0171, low:
61.4257,close: 61.4957,volume: 582741215},
    {x: new Date('2013-03-04'), open: 61.1143, high: 62.2043, low:
59.8571,close: 61.6743,volume: 632856539},
    {x: new Date('2013-03-11'), open: 61.3928, high: 63.4614, low:
60.7343,close: 63.38,volume: 572066981},
```

```

{x: new Date('2013-03-18'), open: 63.0643, high: 66.0143, low:
63.0286,close: 65.9871,volume: 552156035},
{x: new Date('2013-03-25'), open: 66.3843, high: 67.1357, low:
63.0886,close: 63.2371,volume: 390762517},
{x: new Date('2013-04-01'), open: 63.1286, high: 63.3854, low:
59.9543,close: 60.4571,volume: 505273732},
{x: new Date('2013-04-08'), open: 60.6928, high: 62.57, low:
60.3557,close: 61.4,volume: 387323550},
{x: new Date('2013-04-15'), open: 61, high: 61.1271, low: 55.0143,close:
55.79,volume: 709945604},
{x: new Date('2013-04-22'), open: 56.0914, high: 59.8241, low:
55.8964,close: 59.6007,volume: 787007506},
{x: new Date('2013-04-29'), open: 60.0643, high: 64.7471, low: 60,close:
64.2828,volume: 655020017},
{x: new Date('2013-05-06'), open: 65.1014, high: 66.5357, low:
64.3543,close: 64.71,volume: 545488533},
{x: new Date('2013-05-13'), open: 64.5014, high: 65.4143, low:
59.8428,close: 61.8943,volume: 633706550},
{x: new Date('2013-05-20'), open: 61.7014, high: 64.05, low:
61.4428,close: 63.5928,volume: 494379068},
{x: new Date('2013-05-27'), open: 64.2714, high: 65.3, low:
62.7714,close: 64.2478,volume: 362907830},
{x: new Date('2013-06-03'), open: 64.39, high: 64.9186, low:
61.8243,close: 63.1158,volume: 443249793},
{x: new Date('2013-06-10'), open: 63.5328, high: 64.1541, low:
61.2143,close: 61.4357,volume: 389680092},
{x: new Date('2013-06-17'), open: 61.6343, high: 62.2428, low:
58.3,close: 59.0714,volume: 400384818},
{x: new Date('2013-06-24'), open: 58.2, high: 58.38, low: 55.5528,close:
56.6471,volume: 519314826},
{x: new Date('2013-07-01'), open: 57.5271, high: 60.47, low:
57.3171,close: 59.6314,volume: 343878841},
{x: new Date('2013-07-08'), open: 60.0157, high: 61.3986, low:
58.6257,close: 60.93,volume: 384106977},
{x: new Date('2013-07-15'), open: 60.7157, high: 62.1243, low:
60.5957,close: 60.7071,volume: 286035513},
{x: new Date('2013-07-22'), open: 61.3514, high: 63.5128, low:
59.8157,close: 62.9986,volume: 395816827},
{x: new Date('2013-07-29'), open: 62.9714, high: 66.1214, low:
62.8857,close: 66.0771,volume: 339668858},
{x: new Date('2013-08-12'), open: 65.2657, high: 72.0357, low:
65.2328,close: 71.7614,volume: 711563584},
{x: new Date('2013-08-19'), open: 72.0485, high: 73.3914, low:
71.1714,close: 71.5743,volume: 417119660},
{x: new Date('2013-08-26'), open: 71.5357, high: 72.8857, low:
69.4286,close: 69.6023,volume: 392805888},
{x: new Date('2013-09-02'), open: 70.4428, high: 71.7485, low:
69.6214,close: 71.1743,volume: 317244380},
{x: new Date('2013-09-09'), open: 72.1428, high: 72.56, low:
66.3857,close: 66.4143,volume: 669376320},
{x: new Date('2013-09-16'), open: 65.8571, high: 68.3643, low:
63.8886,close: 66.7728,volume: 625142677},
{x: new Date('2013-09-23'), open: 70.8714, high: 70.9871, low:
68.6743,close: 68.9643,volume: 475274537},
{x: new Date('2013-09-30'), open: 68.1786, high: 70.3357, low:
67.773,close: 69.0043,volume: 368198906},

```

```

    {x: new Date('2013-10-07'), open: 69.5086, high: 70.5486, low:
68.3257, close: 70.4017, volume: 361437661},
    {x: new Date('2013-10-14'), open: 69.9757, high: 72.7514, low:
69.9071, close: 72.6985, volume: 342694379},
    {x: new Date('2013-10-21'), open: 73.11, high: 76.1757, low:
72.5757, close: 75.1368, volume: 490458997},
    {x: new Date('2013-10-28'), open: 75.5771, high: 77.0357, low:
73.5057, close: 74.29, volume: 508130174},
    {x: new Date('2013-11-04'), open: 74.4428, high: 75.555, low:
73.1971, close: 74.3657, volume: 318132218},
    {x: new Date('2013-11-11'), open: 74.2843, high: 75.6114, low:
73.4871, close: 74.9987, volume: 306711021},
    {x: new Date('2013-11-18'), open: 74.9985, high: 75.3128, low:
73.3814, close: 74.2571, volume: 282778778},
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        title: 'Months',
        valueType: 'DateTime',
        intervalType: 'Months',
        majorGridLines: { width: 0 },
        crosshairTooltip: { enable: true },
    },
    primaryYAxis: {
        title: 'Price (Million Dollars)',
        minimum: 30, maximum: 180, interval: 30
    },
    axes:[{
        name: 'secondary',
        minimum: 10,
        maximum: 110,
        majorGridLines: { width: 0 },
        opposedPosition: true
    }],
    series:[{
        dataSource: chartData, width: 2, low: 'low', high: 'high', close:
'close', open: 'open',
        xName: 'x', yName: 'y',
        name: 'USA',
        type: 'Candle',
    }],
    indicators: [{
        type: 'Stochastic', field: 'Close', seriesName: 'USA', yAxisName:
'secondary', fill: 'blue',
        kPeriod: 2, dPeriod: 3, showZones: true, periodLine: { color:
'yellow' },
        period: 3, animation: { enable: false }, upperLine: { color: 'red'
}, lowerLine: { color: 'green' }
    }],
    tooltip: { enable: true, shared: true },
    chartArea: { border: { width: 0 } },
    crosshair: { enable: true, lineType: 'Vertical' },
    title: 'AAPL - 2016/2017'
}, '#element');

```

[INDEX.HTML](#)

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Triangular Moving Average (TMA)

Moving average Indicators are used to define the direction of the trend. To render a TMA Indicator, use indicator [type](#) as `Tma` and inject `TmaIndicator` module using `Chart.Inject(TmaIndicator)`.

INDEX.JS

```

var chartData = [
  {x: new Date('2012-10-15'), open: 90.3357, high: 93.2557, low:
87.0885,close: 87.12,volume: 646996264},
  {x: new Date('2012-10-22'), open: 87.4885, high: 90.7685, low:
84.4285,close: 86.2857,volume: 866040680 },
  {x: new Date('2012-10-29'), open: 84.9828, high: 86.1428, low:
82.1071,close: 82.4,volume: 367371310},
  {x: new Date('2012-11-05'), open: 83.3593, high: 84.3914, low:
76.2457,close: 78.1514,volume: 919719846},
  {x: new Date('2012-11-12'), open: 79.1643, high: 79.2143, low:
72.25,close: 75.3825,volume: 894382149},
  {x: new Date('2012-11-19'), open: 77.2443, high: 81.7143, low:
77.1257,close: 81.6428,volume: 527416747},
  {x: new Date('2012-11-26'), open: 82.2714, high: 84.8928, low:
81.7514,close: 83.6114,volume: 646467974},
  {x: new Date('2012-12-03'), open: 84.8071, high: 84.9414, low:
74.09,close: 76.1785,volume: 980096264},
  {x: new Date('2012-12-10'), open: 75, high: 78.5085, low: 72.2257,close:
72.8277,volume: 835016110},
  {x: new Date('2012-12-17'), open: 72.7043, high: 76.4143, low:
71.6043,close: 74.19,volume: 726150329},

```

```

{x: new Date('2012-12-24'), open: 74.3357, high: 74.8928, low:
72.0943,close: 72.7984,volume: 321104733},
{x: new Date('2012-12-31'), open: 72.9328, high: 79.2857, low:
72.7143,close: 75.2857,volume: 540854882},
{x: new Date('2013-01-07'), open: 74.5714, high: 75.9843, low:
73.6,close: 74.3285,volume: 574594262},
{x: new Date('2013-01-14'), open: 71.8114, high: 72.9643, low:
69.0543,close: 71.4285,volume: 803105621},
{x: new Date('2013-01-21'), open: 72.08, high: 73.57, low:
62.1428,close: 62.84,volume: 971912560},
{x: new Date('2013-01-28'), open: 62.5464, high: 66.0857, low:
62.2657,close: 64.8028,volume: 656549587},
{x: new Date('2013-02-04'), open: 64.8443, high: 68.4014, low:
63.1428,close: 67.8543,volume: 743778993},
{x: new Date('2013-02-11'), open: 68.0714, high: 69.2771, low:
65.7028,close: 65.7371,volume: 585292366},
{x: new Date('2013-02-18'), open: 65.8714, high: 66.1043, low:
63.26,close: 64.4014,volume: 421766997},
{x: new Date('2013-02-25'), open: 64.8357, high: 65.0171, low:
61.4257,close: 61.4957,volume: 582741215},
{x: new Date('2013-03-04'), open: 61.1143, high: 62.2043, low:
59.8571,close: 61.6743,volume: 632856539},
{x: new Date('2013-03-11'), open: 61.3928, high: 63.4614, low:
60.7343,close: 63.38,volume: 572066981},
{x: new Date('2013-03-18'), open: 63.0643, high: 66.0143, low:
63.0286,close: 65.9871,volume: 552156035},
{x: new Date('2013-03-25'), open: 66.3843, high: 67.1357, low:
63.0886,close: 63.2371,volume: 390762517},
{x: new Date('2013-04-01'), open: 63.1286, high: 63.3854, low:
59.9543,close: 60.4571,volume: 505273732},
{x: new Date('2013-04-08'), open: 60.6928, high: 62.57, low:
60.3557,close: 61.4,volume: 387323550},
{x: new Date('2013-04-15'), open: 61, high: 61.1271, low: 55.0143,close:
55.79,volume: 709945604},
{x: new Date('2013-04-22'), open: 56.0914, high: 59.8241, low:
55.8964,close: 59.6007,volume: 787007506},
{x: new Date('2013-04-29'), open: 60.0643, high: 64.7471, low: 60,close:
64.2828,volume: 655020017},
{x: new Date('2013-05-06'), open: 65.1014, high: 66.5357, low:
64.3543,close: 64.71,volume: 545488533},
{x: new Date('2013-05-13'), open: 64.5014, high: 65.4143, low:
59.8428,close: 61.8943,volume: 633706550},
{x: new Date('2013-05-20'), open: 61.7014, high: 64.05, low:
61.4428,close: 63.5928,volume: 494379068},
{x: new Date('2013-05-27'), open: 64.2714, high: 65.3, low:
62.7714,close: 64.2478,volume: 362907830},
{x: new Date('2013-06-03'), open: 64.39, high: 64.9186, low:
61.8243,close: 63.1158,volume: 443249793},
{x: new Date('2013-06-10'), open: 63.5328, high: 64.1541, low:
61.2143,close: 61.4357,volume: 389680092},
{x: new Date('2013-06-17'), open: 61.6343, high: 62.2428, low:
58.3,close: 59.0714,volume: 400384818},
{x: new Date('2013-06-24'), open: 58.2, high: 58.38, low: 55.5528,close:
56.6471,volume: 519314826},
{x: new Date('2013-07-01'), open: 57.5271, high: 60.47, low:
57.3171,close: 59.6314,volume: 343878841},

```



```

    {x: new Date('2013-07-08'), open: 60.0157, high: 61.3986, low:
58.6257,close: 60.93,volume: 384106977},
    {x: new Date('2013-07-15'), open: 60.7157, high: 62.1243, low:
60.5957,close: 60.7071,volume: 286035513},
    {x: new Date('2013-07-22'), open: 61.3514, high: 63.5128, low:
59.8157,close: 62.9986,volume: 395816827},
    {x: new Date('2013-07-29'), open: 62.9714, high: 66.1214, low:
62.8857,close: 66.0771,volume: 339668858},
    {x: new Date('2013-08-12'), open: 65.2657, high: 72.0357, low:
65.2328,close: 71.7614,volume: 711563584},
    {x: new Date('2013-08-19'), open: 72.0485, high: 73.3914, low:
71.1714,close: 71.5743,volume: 417119660},
    {x: new Date('2013-08-26'), open: 71.5357, high: 72.8857, low:
69.4286,close: 69.6023,volume: 392805888},
    {x: new Date('2013-09-02'), open: 70.4428, high: 71.7485, low:
69.6214,close: 71.1743,volume: 317244380},
    {x: new Date('2013-09-09'), open: 72.1428, high: 72.56, low:
66.3857,close: 66.4143,volume: 669376320},
    {x: new Date('2013-09-16'), open: 65.8571, high: 68.3643, low:
63.8886,close: 66.7728,volume: 625142677},
    {x: new Date('2013-09-23'), open: 70.8714, high: 70.9871, low:
68.6743,close: 68.9643,volume: 475274537},
    {x: new Date('2013-09-30'), open: 68.1786, high: 70.3357, low:
67.773,close: 69.0043,volume: 368198906},
    {x: new Date('2013-10-07'), open: 69.5086, high: 70.5486, low:
68.3257,close: 70.4017,volume: 361437661},
    {x: new Date('2013-10-14'), open: 69.9757, high: 72.7514, low:
69.9071,close: 72.6985,volume: 342694379},
    {x: new Date('2013-10-21'), open: 73.11, high: 76.1757, low:
72.5757,close: 75.1368,volume: 490458997},
    {x: new Date('2013-10-28'), open: 75.5771, high: 77.0357, low:
73.5057,close: 74.29,volume: 508130174},
    {x: new Date('2013-11-04'), open: 74.4428, high: 75.555, low:
73.1971,close: 74.3657,volume: 318132218},
    {x: new Date('2013-11-11'), open: 74.2843, high: 75.6114, low:
73.4871,close: 74.9987,volume: 306711021},
    {x: new Date('2013-11-18'), open: 74.9985, high: 75.3128, low:
73.3814,close: 74.2571,volume: 282778778},
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        title: 'Months',
        valueType: 'DateTime',
        intervalType: 'Months',
        majorGridLines: { width: 0 },
        crosshairTooltip: { enable: true },
    },
    primaryYAxis: {
        title: 'Price (Million Dollars)',
        minimum: 30, maximum: 180, interval: 30
    },
    axes: [{
        name: 'secondary',
        minimum: 30,
        maximum: 110,
        majorGridLines: { width: 0 },
        opposedPosition: true
    }

```

```

    }],
    series:[{
        dataSource: chartData, width: 2,
        xName: 'x', yName: 'y', low: 'low', high: 'high', close: 'close',
        volume: 'volume', open: 'open',
        name: 'Apple Inc',
        //Series type as RangeColumn
        type: 'Candle', animation: { enable: true }
    }],
    indicators: [{
        type: 'Tma', field: 'Close', seriesName: 'Apple Inc', fill: 'blue',
        period: 3, animation: { enable: true }
    }],
    tooltip: { enable: true, shared: true },
    chartArea: { border: { width: 0 } },
    crosshair: { enable: true, lineType: 'Vertical' },
    title: 'AAPL - 2016/2017'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Customization of Technical Indicators

stroke, stroke-width, and color of signalLine can be customized by using fill, width and dashArray properties. and the period property is used to predict the data forecast calculations. The field value is used to compare the current price with previous price. It is applicable to Bollinger bands and moving

averages. The [showZones](#) property is used to shows/Hides the overBought and overSold regions. It is applicable for RSI and stochastic indicators.

INDEX.JS

```
var chartData = [
    {x: new Date('2012-10-15'), open: 90.3357, high: 93.2557, low: 87.0885, close: 87.12, volume: 646996264},
    {x: new Date('2012-10-22'), open: 87.4885, high: 90.7685, low: 84.4285, close: 86.2857, volume: 866040680 },
    {x: new Date('2012-10-29'), open: 84.9828, high: 86.1428, low: 82.1071, close: 82.4, volume: 367371310},
    {x: new Date('2012-11-05'), open: 83.3593, high: 84.3914, low: 76.2457, close: 78.1514, volume: 919719846},
    {x: new Date('2012-11-12'), open: 79.1643, high: 79.2143, low: 72.25, close: 75.3825, volume: 894382149},
    {x: new Date('2012-11-19'), open: 77.2443, high: 81.7143, low: 77.1257, close: 81.6428, volume: 527416747},
    {x: new Date('2012-11-26'), open: 82.2714, high: 84.8928, low: 81.7514, close: 83.6114, volume: 646467974},
    {x: new Date('2012-12-03'), open: 84.8071, high: 84.9414, low: 74.09, close: 76.1785, volume: 980096264},
    {x: new Date('2012-12-10'), open: 75, high: 78.5085, low: 72.2257, close: 72.8277, volume: 835016110},
    {x: new Date('2012-12-17'), open: 72.7043, high: 76.4143, low: 71.6043, close: 74.19, volume: 726150329},
    {x: new Date('2012-12-24'), open: 74.3357, high: 74.8928, low: 72.0943, close: 72.7984, volume: 321104733},
    {x: new Date('2012-12-31'), open: 72.9328, high: 79.2857, low: 72.7143, close: 75.2857, volume: 540854882},
    {x: new Date('2013-01-07'), open: 74.5714, high: 75.9843, low: 73.6, close: 74.3285, volume: 574594262},
    {x: new Date('2013-01-14'), open: 71.8114, high: 72.9643, low: 69.0543, close: 71.4285, volume: 803105621},
    {x: new Date('2013-01-21'), open: 72.08, high: 73.57, low: 62.1428, close: 62.84, volume: 971912560},
    {x: new Date('2013-01-28'), open: 62.5464, high: 66.0857, low: 62.2657, close: 64.8028, volume: 656549587},
    {x: new Date('2013-02-04'), open: 64.8443, high: 68.4014, low: 63.1428, close: 67.8543, volume: 743778993},
    {x: new Date('2013-02-11'), open: 68.0714, high: 69.2771, low: 65.7028, close: 65.7371, volume: 585292366},
    {x: new Date('2013-02-18'), open: 65.8714, high: 66.1043, low: 63.26, close: 64.4014, volume: 421766997},
    {x: new Date('2013-02-25'), open: 64.8357, high: 65.0171, low: 61.4257, close: 61.4957, volume: 582741215},
    {x: new Date('2013-03-04'), open: 61.1143, high: 62.2043, low: 59.8571, close: 61.6743, volume: 632856539},
    {x: new Date('2013-03-11'), open: 61.3928, high: 63.4614, low: 60.7343, close: 63.38, volume: 572066981},
    {x: new Date('2013-03-18'), open: 63.0643, high: 66.0143, low: 63.0286, close: 65.9871, volume: 552156035},
    {x: new Date('2013-03-25'), open: 66.3843, high: 67.1357, low: 63.0886, close: 63.2371, volume: 390762517},
    {x: new Date('2013-04-01'), open: 63.1286, high: 63.3854, low: 59.9543, close: 60.4571, volume: 505273732},
```

```

{x: new Date('2013-04-08'), open: 60.6928, high: 62.57, low:
60.3557,close: 61.4,volume: 387323550},
{x: new Date('2013-04-15'), open: 61, high: 61.1271, low: 55.0143,close:
55.79,volume: 709945604},
{x: new Date('2013-04-22'), open: 56.0914, high: 59.8241, low:
55.8964,close: 59.6007,volume: 787007506},
{x: new Date('2013-04-29'), open: 60.0643, high: 64.7471, low: 60,close:
64.2828,volume: 655020017},
{x: new Date('2013-05-06'), open: 65.1014, high: 66.5357, low:
64.3543,close: 64.71,volume: 545488533},
{x: new Date('2013-05-13'), open: 64.5014, high: 65.4143, low:
59.8428,close: 61.8943,volume: 633706550},
{x: new Date('2013-05-20'), open: 61.7014, high: 64.05, low:
61.4428,close: 63.5928,volume: 494379068},
{x: new Date('2013-05-27'), open: 64.2714, high: 65.3, low:
62.7714,close: 64.2478,volume: 362907830},
{x: new Date('2013-06-03'), open: 64.39, high: 64.9186, low:
61.8243,close: 63.1158,volume: 443249793},
{x: new Date('2013-06-10'), open: 63.5328, high: 64.1541, low:
61.2143,close: 61.4357,volume: 389680092},
{x: new Date('2013-06-17'), open: 61.6343, high: 62.2428, low:
58.3,close: 59.0714,volume: 400384818},
{x: new Date('2013-06-24'), open: 58.2, high: 58.38, low: 55.5528,close:
56.6471,volume: 519314826},
{x: new Date('2013-07-01'), open: 57.5271, high: 60.47, low:
57.3171,close: 59.6314,volume: 343878841},
{x: new Date('2013-07-08'), open: 60.0157, high: 61.3986, low:
58.6257,close: 60.93,volume: 384106977},
{x: new Date('2013-07-15'), open: 60.7157, high: 62.1243, low:
60.5957,close: 60.7071,volume: 286035513},
{x: new Date('2013-07-22'), open: 61.3514, high: 63.5128, low:
59.8157,close: 62.9986,volume: 395816827},
{x: new Date('2013-07-29'), open: 62.9714, high: 66.1214, low:
62.8857,close: 66.0771,volume: 339668858},
{x: new Date('2013-08-12'), open: 65.2657, high: 72.0357, low:
65.2328,close: 71.7614,volume: 711563584},
{x: new Date('2013-08-19'), open: 72.0485, high: 73.3914, low:
71.1714,close: 71.5743,volume: 417119660},
{x: new Date('2013-08-26'), open: 71.5357, high: 72.8857, low:
69.4286,close: 69.6023,volume: 392805888},
{x: new Date('2013-09-02'), open: 70.4428, high: 71.7485, low:
69.6214,close: 71.1743,volume: 317244380},
{x: new Date('2013-09-09'), open: 72.1428, high: 72.56, low:
66.3857,close: 66.4143,volume: 669376320},
{x: new Date('2013-09-16'), open: 65.8571, high: 68.3643, low:
63.8886,close: 66.7728,volume: 625142677},
{x: new Date('2013-09-23'), open: 70.8714, high: 70.9871, low:
68.6743,close: 68.9643,volume: 475274537},
{x: new Date('2013-09-30'), open: 68.1786, high: 70.3357, low:
67.773,close: 69.0043,volume: 368198906},
{x: new Date('2013-10-07'), open: 69.5086, high: 70.5486, low:
68.3257,close: 70.4017,volume: 361437661},
{x: new Date('2013-10-14'), open: 69.9757, high: 72.7514, low:
69.9071,close: 72.6985,volume: 342694379},
{x: new Date('2013-10-21'), open: 73.11, high: 76.1757, low:
72.5757,close: 75.1368,volume: 490458997},

```

```

    {x: new Date('2013-10-28'), open: 75.5771, high: 77.0357, low:
73.5057,close: 74.29,volume: 508130174},
    {x: new Date('2013-11-04'), open: 74.4428, high: 75.555, low:
73.1971,close: 74.3657,volume: 318132218},
    {x: new Date('2013-11-11'), open: 74.2843, high: 75.6114, low:
73.4871,close: 74.9987,volume: 306711021},
    {x: new Date('2013-11-18'), open: 74.9985, high: 75.3128, low:
73.3814,close: 74.2571,volume: 282778778},
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    title: 'Months',
    valueType: 'DateTime',
    intervalType: 'Months',
    majorGridLines: { width: 0 },
    crosshairTooltip: { enable: true },
  },
  primaryYAxis: {
    title: 'Price (Million Dollars)',
    minimum: 30, maximum: 180, interval: 30
  },
  axes: [{
    name: 'secondary',
    minimum: 30,
    maximum: 110,
    majorGridLines: { width: 0 },
    opposedPosition: true
  }],
  series:[{
    dataSource: chartData, width: 2,
    xName: 'x', yName: 'y', low: 'low', high: 'high', close: 'close',
    volume: 'volume', open: 'open',
    name: 'Apple Inc',
    //Series type as RangeColumn
    type: 'Candle', animation: { enable: true }
  }],
  indicators: [{
    type: 'Tma', field: 'Low', seriesName: 'Apple Inc', fill: 'red',
    period: 3, animation: { enable: true }
  }],
  tooltip: { enable: true, shared: true },
  chartArea: { border: { width: 0 } },
  crosshair: { enable: true, lineType: 'Vertical' },
  title: 'AAPL - 2016/2017'
}, '#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Data Source

Usually technical indicators are added along with a financial series. The [seriesName](#) represents the series, the data of which has to be analysed through indicators.

Technical indicators can also be added without series using [dataSource](#) property of indicator.

INDEX.JS

```

var chartData = [
    { x: new Date('2012-10-15'), open: 90.3357, high: 93.2557, low: 87.0885,
close: 87.12, volume: 646996264 },
    { x: new Date('2012-10-22'), open: 87.4885, high: 90.7685, low: 84.4285,
close: 86.2857, volume: 866040680 },
    { x: new Date('2012-10-29'), open: 84.9828, high: 86.1428, low: 82.1071,
close: 82.4, volume: 367371310 },
    { x: new Date('2012-11-05'), open: 83.3593, high: 84.3914, low: 76.2457,
close: 78.1514, volume: 919719846 },
    { x: new Date('2012-11-12'), open: 79.1643, high: 79.2143, low: 72.25,
close: 75.3825, volume: 894382149 },
    { x: new Date('2012-11-19'), open: 77.2443, high: 81.7143, low: 77.1257,
close: 81.6428, volume: 527416747 },
    { x: new Date('2012-11-26'), open: 82.2714, high: 84.8928, low: 81.7514,
close: 83.6114, volume: 646467974 },
    { x: new Date('2012-12-03'), open: 84.8071, high: 84.9414, low: 74.09,
close: 76.1785, volume: 980096264 },
    { x: new Date('2012-12-10'), open: 75, high: 78.5085, low: 72.2257,
close: 72.8277, volume: 835016110 },
    { x: new Date('2012-12-17'), open: 72.7043, high: 76.4143, low: 71.6043,
close: 74.19, volume: 726150329 },
    { x: new Date('2012-12-24'), open: 74.3357, high: 74.8928, low: 72.0943,
close: 72.7984, volume: 321104733 },
    { x: new Date('2012-12-31'), open: 72.9328, high: 79.2857, low: 72.7143,
close: 75.2857, volume: 540854882 },
    { x: new Date('2013-01-07'), open: 74.5714, high: 75.9843, low: 73.6,
close: 74.3285, volume: 574594262 },

```

```

{ x: new Date('2013-01-14'), open: 71.8114, high: 72.9643, low: 69.0543,
close: 71.4285, volume: 803105621 },
{ x: new Date('2013-01-21'), open: 72.08, high: 73.57, low: 62.1428,
close: 62.84, volume: 971912560 },
{ x: new Date('2013-01-28'), open: 62.5464, high: 66.0857, low: 62.2657,
close: 64.8028, volume: 656549587 },
{ x: new Date('2013-02-04'), open: 64.8443, high: 68.4014, low: 63.1428,
close: 67.8543, volume: 743778993 },
{ x: new Date('2013-02-11'), open: 68.0714, high: 69.2771, low: 65.7028,
close: 65.7371, volume: 585292366 },
{ x: new Date('2013-02-18'), open: 65.8714, high: 66.1043, low: 63.26,
close: 64.4014, volume: 421766997 },
{ x: new Date('2013-02-25'), open: 64.8357, high: 65.0171, low: 61.4257,
close: 61.4957, volume: 582741215 },
{ x: new Date('2013-03-04'), open: 61.1143, high: 62.2043, low: 59.8571,
close: 61.6743, volume: 632856539 },
{ x: new Date('2013-03-11'), open: 61.3928, high: 63.4614, low: 60.7343,
close: 63.38, volume: 572066981 },
{ x: new Date('2013-03-18'), open: 63.0643, high: 66.0143, low: 63.0286,
close: 65.9871, volume: 552156035 },
{ x: new Date('2013-03-25'), open: 66.3843, high: 67.1357, low: 63.0886,
close: 63.2371, volume: 390762517 },
{ x: new Date('2013-04-01'), open: 63.1286, high: 63.3854, low: 59.9543,
close: 60.4571, volume: 505273732 },
{ x: new Date('2013-04-08'), open: 60.6928, high: 62.57, low: 60.3557,
close: 61.4, volume: 387323550 },
{ x: new Date('2013-04-15'), open: 61, high: 61.1271, low: 55.0143,
close: 55.79, volume: 709945604 },
{ x: new Date('2013-04-22'), open: 56.0914, high: 59.8241, low: 55.8964,
close: 59.6007, volume: 787007506 },
{ x: new Date('2013-04-29'), open: 60.0643, high: 64.7471, low: 60,
close: 64.2828, volume: 655020017 },
{ x: new Date('2013-05-06'), open: 65.1014, high: 66.5357, low: 64.3543,
close: 64.71, volume: 545488533 },
{ x: new Date('2013-05-13'), open: 64.5014, high: 65.4143, low: 59.8428,
close: 61.8943, volume: 633706550 },
{ x: new Date('2013-05-20'), open: 61.7014, high: 64.05, low: 61.4428,
close: 63.5928, volume: 494379068 },
{ x: new Date('2013-05-27'), open: 64.2714, high: 65.3, low: 62.7714,
close: 64.2478, volume: 362907830 },
{ x: new Date('2013-06-03'), open: 64.39, high: 64.9186, low: 61.8243,
close: 63.1158, volume: 443249793 },
{ x: new Date('2013-06-10'), open: 63.5328, high: 64.1541, low: 61.2143,
close: 61.4357, volume: 389680092 },
{ x: new Date('2013-06-17'), open: 61.6343, high: 62.2428, low: 58.3,
close: 59.0714, volume: 400384818 },
{ x: new Date('2013-06-24'), open: 58.2, high: 58.38, low: 55.5528,
close: 56.6471, volume: 519314826 },
{ x: new Date('2013-07-01'), open: 57.5271, high: 60.47, low: 57.3171,
close: 59.6314, volume: 343878841 },
{ x: new Date('2013-07-08'), open: 60.0157, high: 61.3986, low: 58.6257,
close: 60.93, volume: 384106977 },
{ x: new Date('2013-07-15'), open: 60.7157, high: 62.1243, low: 60.5957,
close: 60.7071, volume: 286035513 },
{ x: new Date('2013-07-22'), open: 61.3514, high: 63.5128, low: 59.8157,
close: 62.9986, volume: 395816827 },

```

```

    { x: new Date('2013-07-29'), open: 62.9714, high: 66.1214, low: 62.8857,
      close: 66.0771, volume: 339668858 },
    { x: new Date('2013-08-12'), open: 65.2657, high: 72.0357, low: 65.2328,
      close: 71.7614, volume: 711563584 },
    { x: new Date('2013-08-19'), open: 72.0485, high: 73.3914, low: 71.1714,
      close: 71.5743, volume: 417119660 },
    { x: new Date('2013-08-26'), open: 71.5357, high: 72.8857, low: 69.4286,
      close: 69.6023, volume: 392805888 },
    { x: new Date('2013-09-02'), open: 70.4428, high: 71.7485, low: 69.6214,
      close: 71.1743, volume: 317244380 },
    { x: new Date('2013-09-09'), open: 72.1428, high: 72.56, low: 66.3857,
      close: 66.4143, volume: 669376320 },
    { x: new Date('2013-09-16'), open: 65.8571, high: 68.3643, low: 63.8886,
      close: 66.7728, volume: 625142677 },
    { x: new Date('2013-09-23'), open: 70.8714, high: 70.9871, low: 68.6743,
      close: 68.9643, volume: 475274537 },
    { x: new Date('2013-09-30'), open: 68.1786, high: 70.3357, low: 67.773,
      close: 69.0043, volume: 368198906 },
    { x: new Date('2013-10-07'), open: 69.5086, high: 70.5486, low: 68.3257,
      close: 70.4017, volume: 361437661 },
    { x: new Date('2013-10-14'), open: 69.9757, high: 72.7514, low: 69.9071,
      close: 72.6985, volume: 342694379 },
    { x: new Date('2013-10-21'), open: 73.11, high: 76.1757, low: 72.5757,
      close: 75.1368, volume: 490458997 },
    { x: new Date('2013-10-28'), open: 75.5771, high: 77.0357, low: 73.5057,
      close: 74.29, volume: 508130174 },
    { x: new Date('2013-11-04'), open: 74.4428, high: 75.555, low: 73.1971,
      close: 74.3657, volume: 318132218 },
    { x: new Date('2013-11-11'), open: 74.2843, high: 75.6114, low: 73.4871,
      close: 74.9987, volume: 306711021 },
    { x: new Date('2013-11-18'), open: 74.9985, high: 75.3128, low: 73.3814,
      close: 74.2571, volume: 282778778 },
  ];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    title: 'Months',
    valueType: 'DateTime',
    intervalType: 'Months',
    majorGridLines: { width: 0 },
    crosshairTooltip: { enable: true },
  },
  primaryYAxis: {
    title: 'Price (Million Dollars)',
    minimum: 30,
    maximum: 180,
    interval: 30,
  },
  axes: [{
    name: 'secondary',
    minimum: -7000000000, maximum: 5000000000,
    interval: 6000000000,
    majorGridLines: { width: 0 },
    opposedPosition: true
  }],
  indicators: [{
    type: 'AccumulationDistribution',

```



```

        dataSource: chartData, low: 'low', high: 'high', close: 'close',
        volume: 'volume', xName: 'x',
        yAxisName: 'secondary', fill: 'blue', open: 'open',
        period: 3, animation: { enable: true }
    }],
    tooltip: { enable: true, shared: true },
    chartArea: { border: { width: 0 } },
    axisLabelRender: (args) => {
        if (args.axis.name === 'secondary') {
            var value = (args.text) / 1000000000;
            args.text = (value) + 'bn';
        }
    },
    crosshair: { enable: true, lineType: 'Vertical' },
    title: 'AAPL - 2016/2017'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

<!-- markdownlint-disable MD036 -->

Trend lines in ##Platform_Name## Chart control

Trendlines are used to show the direction and speed of price.

Trendlines can be generated for Cartesian type series (Line, Column, Scatter, Area, Candle, Hilo etc.) except bar type series. You can add more than one trendline to a series.

Chart supports 6 types of trendlines.

Linear

A linear trendline is a best fit straight line that is used with simpler data sets. To render a linear trendline, use trendline [type](#) as `Linear` and inject `Trendlines` module using `Chart.Inject(Trendlines)`.

INDEX.JS

```
var data = [];
var yValue = [7.66, 8.03, 8.41, 8.97, 8.77, 8.20, 8.16, 7.89, 8.68, 9.48,
10.11, 11.36, 12.34, 12.60, 12.95, 13.91, 16.21, 17.50, 22.72, 28.14, 31.26,
31.39, 32.43, 35.52, 36.36,
41.33, 43.12, 45.00, 47.23, 48.62, 46.60, 45.28, 44.01, 45.17, 41.20,
43.41, 48.32, 45.65, 46.61, 53.34, 58.53];
var point;
var i; var j = 0;
for (i = 1973; i <= 2013; i++) {
    point = { x: i, y: yValue[j] };
    data.push(point);
    j++;
}
var chart = new ej.charts.Chart({
    primaryXAxis: {
        title: 'Months',
    },
    primaryYAxis: {
        title: 'Rupees against Dollars',
        interval: 5
    },
    tooltip: { enable: true },
    chartArea: { border: { width: 0 } },
    series: [{
        dataSource: data,
        xName: 'x', yName: 'y',
        name: 'Apple Inc',
        fill: '#0066FF',
        //Series type as scatter
        type: 'Scatter',
        trendlines: [{ type: 'Linear' }]
    }],
    title: 'Historical Indian Rupee Rate (INR USD)'
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Exponential

An exponential trendline is a curved line that is most useful when data values rise or fall at increasingly higher rates. You cannot create an exponential trendline, if your data contains zero or negative values.

To render an exponential trendline, use trendline [type](#) as **Exponential** and inject **Trendlines** module using **Chart.Inject(Trendlines)**.

INDEX.JS

```

var data = [];
var yValue = [7.66, 8.03, 8.41, 8.97, 8.77, 8.20, 8.16, 7.89, 8.68, 9.48,
10.11, 11.36, 12.34, 12.60, 12.95, 13.91, 16.21, 17.50, 22.72, 28.14, 31.26,
31.39, 32.43, 35.52, 36.36,
41.33, 43.12, 45.00, 47.23, 48.62, 46.60, 45.28, 44.01, 45.17, 41.20,
43.41, 48.32, 45.65, 46.61, 53.34, 58.53];
var point;
var i; var j = 0;
for (i = 1973; i <= 2013; i++) {
    point = { x: i, y: yValue[j] };
    data.push(point);
    j++;
}
var chart = new ej.charts.Chart({
    primaryXAxis: {
        title: 'Months',
    },
    primaryYAxis: {
        title: 'Rupees against Dollars',
        interval: 5
    },
    tooltip: { enable: true },
    chartArea: { border: { width: 0 } },
    series: [{
        dataSource: data,
        xName: 'x', yName: 'y',
        name: 'Apple Inc',
        fill: '#0066FF',
        type: 'Scatter',
        trendlines: [

```

```

        { type: 'Exponential',enableTooltip:true,marker:{visible:true} }
    ],
    title: 'Historical Indian Rupee Rate (INR USD) '
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Logarithmic

A logarithmic trendline is a best-fit curved line that is most useful when the rate of change in the data increases or decreases quickly and then levels out. A logarithmic trendline can use negative and/or positive values.

To render a logarithmic trendline, use trendline [type](#) as `Logarithmic` and inject `Trendlines` module using `Chart.Inject(Trendlines)`.

INDEX.JS

```

var data = [];
var yValue = [7.66, 8.03, 8.41, 8.97, 8.77, 8.20, 8.16, 7.89, 8.68, 9.48,
10.11, 11.36, 12.34, 12.60, 12.95, 13.91, 16.21, 17.50, 22.72, 28.14, 31.26,
31.39, 32.43, 35.52, 36.36,
41.33, 43.12, 45.00, 47.23, 48.62, 46.60, 45.28, 44.01, 45.17, 41.20,
43.41, 48.32, 45.65, 46.61, 53.34, 58.53];
var point;
var i; var j = 0;
for (i = 1973; i <= 2013; i++) {

```

```

        point = { x: i, y: yValue[j] };
        data.push(point);
        j++;
    }
    var chart = new ej.charts.Chart({
        primaryXAxis: {
            title: 'Months',
        },
        primaryYAxis: {
            title: 'Rupees against Dollars',
            interval: 5
        },
        tooltip: { enable: true },
        chartArea: { border: { width: 0 } },
        series: [{
            dataSource: data,
            xName: 'x', yName: 'y',
            name: 'Apple Inc',
            fill: '#0066FF',
            //Series type as scatter
            type: 'Scatter',
            trendlines: [{ type: 'Logarithmic' }]
        }],
        title: 'Historical Indian Rupee Rate (INR USD) '
    }, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
    <script>
    var ele = document.getElementById('container');
    if(ele) {
        ele.style.visibility = "visible";
    }
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

Polynomial

A polynomial trendline is a curved line that is used when data fluctuates.

To render a polynomial trendline, use trendline [type](#) as **Polynomial** and inject **Trendlines** module using **Chart.Inject(Trendlines)**.

polynomialOrder used to define the polynomial value.

INDEX.JS

```
var data = [];
var yValue = [7.66, 8.03, 8.41, 8.97, 8.77, 8.20, 8.16, 7.89, 8.68, 9.48,
10.11, 11.36, 12.34, 12.60, 12.95, 13.91, 16.21, 17.50, 22.72, 28.14, 31.26,
31.39, 32.43, 35.52, 36.36,
41.33, 43.12, 45.00, 47.23, 48.62, 46.60, 45.28, 44.01, 45.17, 41.20,
43.41, 48.32, 45.65, 46.61, 53.34, 58.53];
var point;
var i; var j = 0;
for (i = 1973; i <= 2013; i++) {
    point = { x: i, y: yValue[j] };
    data.push(point);
    j++;
}
var chart = new ej.charts.Chart({
    primaryXAxis: {
        title: 'Months',
    },
    primaryYAxis: {
        title: 'Rupees against Dollars',
        interval: 5
    },
    tooltip: { enable: true },
    chartArea: { border: { width: 0 } },
    series: [{
        dataSource: data,
        xName: 'x', yName: 'y',
        name: 'Apple Inc',
        fill: '#0066FF',
        //Series type as scatter
        type: 'Scatter',
        trendlines: [{ type: 'Polynomial' }]
    }],
    title: 'Historical Indian Rupee Rate (INR USD)'
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
<title>EJ2 Animation</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Power

A power trendline is a curved line that is best used with data sets that compare measurements that increase at a specific rate.

To render a power trendline, use trendline [type](#) as **Power** and inject **Trendlines** module using `Chart.Inject(Trendlines)`.

INDEX.JS

```

var powerData= [
    { x: 1, y: 10 }, { x: 2, y: 50 }, { x: 3, y: 80 }, { x: 4, y: 110 },
    { x: 5, y: 180 }, { x: 6, y: 220 }, { x: 7, y: 300 }, { x: 8, y: 370 },
    { x: 9, y: 490 }, { x: 10, y: 500 }
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        title: 'Months',
    },
    primaryYAxis: {
        title: 'Rupees against Dollars',
        interval: 50
    },
    tooltip:{enable:true},
    chartArea: { border: { width: 0 } },
    series: [{
        dataSource: powerData,
        xName: 'x', yName: 'y',
        name: 'Apple Inc',
        fill: '#0066FF',
        //Series type as scatter
        type: 'Scatter',
        trendlines: [{ type: 'Power' }]
    }],
    title: 'Historical Indian Rupee Rate (INR USD) ',
}, '#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Moving Average

A moving average trendline smoothen out fluctuations in data to show a pattern or trend more clearly.

To render a moving average trendline, use trendline [type](#) as `MovingAverage` and inject `Trendlines` module using `Chart.Inject(Trendlines)`.

`period` property defines the period to find the moving average.

INDEX.JS

```

var data = [];
var yValue = [7.66, 8.03, 8.41, 8.97, 8.77, 8.20, 8.16, 7.89, 8.68, 9.48,
10.11, 11.36, 12.34, 12.60, 12.95, 13.91, 16.21, 17.50, 22.72, 28.14, 31.26,
31.39, 32.43, 35.52, 36.36,
41.33, 43.12, 45.00, 47.23, 48.62, 46.60, 45.28, 44.01, 45.17, 41.20,
43.41, 48.32, 45.65, 46.61, 53.34, 58.53];
var point;
var i; var j = 0;
for (i = 1973; i <= 2013; i++) {
  point = { x: i, y: yValue[j] };
  data.push(point);
  j++;
}
var chart = new ej.charts.Chart({
  primaryXAxis: {
    title: 'Months',

```



```

    },
    primaryYAxis: {
        title: 'Rupees against Dollars',
        interval: 5
    },
    tooltip: { enable: true },
    chartArea: { border: { width: 0 } },
    series: [{
        dataSource: data,
        xName: 'x', yName: 'y',
        name: 'Apple Inc',
        fill: '#0066FF',
        //Series type as scatter
        type: 'Scatter',
        trendlines: [{ type: 'MovingAverage' }]
    }],
    title: 'Historical Indian Rupee Rate (INR USD) '
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Customization of Trendline

The [fill](#) and [width](#) properties are used to customize the appearance of the trendline.

INDEX.JS

```
var data = [];
```

```

var yValue = [7.66, 8.03, 8.41, 8.97, 8.77, 8.20, 8.16, 7.89, 8.68, 9.48,
10.11, 11.36, 12.34, 12.60, 12.95, 13.91, 16.21, 17.50, 22.72, 28.14, 31.26,
31.39, 32.43, 35.52, 36.36,
41.33, 43.12, 45.00, 47.23, 48.62, 46.60, 45.28, 44.01, 45.17, 41.20,
43.41, 48.32, 45.65, 46.61, 53.34, 58.53];
var point;
var i; var j = 0;
for (i = 1973; i <= 2013; i++) {
    point = { x: i, y: yValue[j] };
    data.push(point);
    j++;
}
var chart = new ej.charts.Chart({
    primaryXAxis: {
        title: 'Months',
    },
    primaryYAxis: {
        title: 'Rupees against Dollars',
        interval: 5
    },
    tooltip:{enable:true},
    chartArea: { border: { width: 0 } },
    series: [{
        dataSource: data,
        xName: 'x', yName: 'y',
        name: 'Apple Inc',
        fill: '#0066FF',
        //Series type as scatter
        type: 'Scatter',
        trendlines: [{ type: 'MovingAverage', fill: 'red', width:2 }]
    }],
    title: 'Historical Indian Rupee Rate (INR USD) '
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
</body>
</html>

```

```

var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Forecasting

Trendline forecasting is the prediction of future/past situations.

Forward Forecasting and Backward Forecasting are the two types of forecasting.

Forward Forecasting

The value set for forwardForecast is used to determine the distance moving towards the future trend.

INDEX.JS

```

var data = [];
var yValue = [7.66, 8.03, 8.41, 8.97, 8.77, 8.20, 8.16, 7.89, 8.68, 9.48,
10.11, 11.36, 12.34, 12.60, 12.95, 13.91, 16.21, 17.50, 22.72, 28.14, 31.26,
31.39, 32.43, 35.52, 36.36,
41.33, 43.12, 45.00, 47.23, 48.62, 46.60, 45.28, 44.01, 45.17, 41.20,
43.41, 48.32, 45.65, 46.61, 53.34, 58.53];
var point;
var i; var j = 0;
for (i = 1973; i <= 2013; i++) {
    point = { x: i, y: yValue[j] };
    data.push(point);
    j++;
}
var chart = new ej.charts.Chart({
    primaryXAxis: {
        title: 'Months',
    },
    primaryYAxis: {
        title: 'Rupees against Dollars',
        interval: 5
    },
    tooltip: { enable: true },
    chartArea: { border: { width: 0 } },
    series: [{
        dataSource: data,
        xName: 'x', yName: 'y',
        name: 'Apple Inc',
        fill: '#0066FF',
        //Series type as scatter
        type: 'Scatter',
        trendlines: [{
            type: 'Linear',
            //forward forecast value
            forwardForecast: 5
        }]
    }],
    title: 'Historical Indian Rupee Rate (INR USD)'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Backward Forecasting

The value set for the backwardForecast is used to determine the past trends.

INDEX.JS

```

var data = [];
var yValue = [7.66, 8.03, 8.41, 8.97, 8.77, 8.20, 8.16, 7.89, 8.68, 9.48,
10.11, 11.36, 12.34, 12.60, 12.95, 13.91, 16.21, 17.50, 22.72, 28.14, 31.26,
31.39, 32.43, 35.52, 36.36,
41.33, 43.12, 45.00, 47.23, 48.62, 46.60, 45.28, 44.01, 45.17, 41.20,
43.41, 48.32, 45.65, 46.61, 53.34, 58.53];
var point;
var i; var j = 0;
for (i = 1973; i <= 2013; i++) {
  point = { x: i, y: yValue[j] };
  data.push(point);
  j++;
}
var chart = new ej.charts.Chart({
  primaryXAxis: {
    title: 'Months',
  },
  primaryYAxis: {
    title: 'Rupees against Dollars',

```

```

        interval: 5
    },
    tooltip: { enable: true },
    chartArea: { border: { width: 0 } },
    series: [{
        dataSource: data,
        xName: 'x', yName: 'y',
        name: 'Apple Inc',
        fill: '#0066FF',
        //Series type as scatter
        type: 'Scatter',
        trendlines: [{
            type: 'Linear',
            //backward forecast value
            backwardForecast: 5
        }]
    }],
    title: 'Historical Indian Rupee Rate (INR USD) '
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Show or hide a trendline

You can show or hide the trendline by setting trendline `visible` property.

INDEX.JS

```

var data = [];
var yValue = [7.66, 8.03, 8.41, 8.97, 8.77, 8.20, 8.16, 7.89, 8.68, 9.48,
10.11, 11.36, 12.34, 12.60, 12.95, 13.91, 16.21, 17.50, 22.72, 28.14, 31.26,
31.39, 32.43, 35.52, 36.36,
41.33, 43.12, 45.00, 47.23, 48.62, 46.60, 45.28, 44.01, 45.17, 41.20,
43.41, 48.32, 45.65, 46.61, 53.34, 58.53];
var point;
var i; var j = 0;
for (i = 1973; i <= 2013; i++) {
    point = { x: i, y: yValue[j] };
    data.push(point);
    j++;
}
var chart = new ej.charts.Chart({
    primaryXAxis: {
        title: 'Months',
    },
    primaryYAxis: {
        title: 'Rupees against Dollars',
        interval: 5
    },
    tooltip:{enable:true},
    chartArea: { border: { width: 0 } },
    series: [{
        dataSource: data,
        xName: 'x', yName: 'y',
        name: 'Apple Inc',
        fill: '#0066FF',
        //Series type as scatter
        type: 'Scatter',
        trendlines: [{ visible: false }]
    }],
    title: 'Historical Indian Rupee Rate (INR USD) '
}, '#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
```

```
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Data markers in ##Platform_Name## Chart control

Data markers are used to provide information about the data points in the series. You can add a shape to adorn each data point.

<!-- markdownlint-disable MD036 -->

Marker

<!-- markdownlint-disable MD036 -->

Markers can be added to points by enabling the [visible](#) option of the marker property. By default, distinct markers will be enabled for each series in the chart.

INDEX.JS

```
var numData = [{ x: 2005, y: 28, y1: 18 }, { x: 2006, y: 25, y1: 10 }, { x:
2007, y: 26, y1: 20 }, { x: 2008, y: 47, y1: 35 },
{ x: 2009, y: 32, y1: 23 }, { x: 2010, y: 35, y1: 25 }, { x: 2011, y:
30, y1: 15 }];
var chart = new ej.charts.Chart({
    series: [
        {
            type: 'Line',
            dataSource: numData, xName: 'x', yName: 'y',
            marker: {
                visible: true
            }
        },
        {
            type: 'Line',
            dataSource: numData, xName: 'x', yName: 'y1',
            marker: {
                visible: true
            }
        }
    ],
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
<title>EJ2 Animation</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Shape

Markers can be assigned with different shapes such as Rectangle, Circle, Diamond, etc. using the [shape](#) property.

INDEX.JS

```

var chartData = [
    { x: 'WW', y: 12, y1: 22, y2: 38.3, y3: 50 },
    { x: 'EU', y: 9.9, y1: 26, y2: 45.2, y3: 63.6 },
    { x: 'APAC', y: 4.4, y1: 9.3, y2: 18.2, y3: 20.9 },
    { x: 'LATAM', y: 6.4, y1: 28, y2: 46.7, y3: 65.1 },
    { x: 'MEA', y: 30, y1: 45.7, y2: 61.5, y3: 73 },
    { x: 'NA', y: 25.3, y1: 35.9, y2: 64, y3: 81.4 }
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        title: 'Countries',
        valueType: 'Category', interval: 1,
        labelIntersectAction : 'Rotate45'
    },
    primaryYAxis:
    {
        title: 'Penetration (%)',
        labelFormat: '{value}%',
        minimum: 0, maximum: 90
    },
    series: [
        {
            type: 'Line', name: 'December 2007',
            dataSource: chartData, xName: 'x', yName: 'y', width: 2,
            marker: {
                visible: true,
                width: 10, height: 10,
                shape: 'Diamond'
            }
        }, {

```



```

        type: 'Line', name: 'December 2008',
        dataSource: chartData, xName: 'x', yName: 'y1', width: 2,
        //Marker for chart series
        marker: {
            visible: true,
            width: 10, height: 10,
            shape: 'Pentagon'
        }
    }, {
        type: 'Line', name: 'December 2009',
        dataSource: chartData, xName: 'x', yName: 'y2', width: 2,
        marker: {
            visible: true,
            width: 10, height: 10,
            shape: 'Triangle',
        }
    }, {
        type: 'Line', name: 'December 2010',
        dataSource: chartData, xName: 'x', yName: 'y3', width: 2,
        marker: {
            visible: true,
            width: 10, height: 10,
            shape: 'Circle'
        }
    }
],
title: 'FB Penetration of Internet Audience'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
    <script>
    var ele = document.getElementById('container');
    if(ele) {
        ele.style.visibility = "visible";
    }
    </script>

```

```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Note : To know more about the marker shape type refer the [shape](#).

Images

Apart from the shapes, you can also add custom images to mark the data point using the [imageUrl](#) property.

INDEX.JS

```
var chartData = [
  { x: "Jan", y: 60 }, { x: "Feb", y: 50 },
  { x: "Mar", y: 64 }, { x: "Apr", y: 63 },
  { x: "May", y: 81 }, { x: "Jun", y: 64 },
  { x: "Jul", y: 82 }, { x: "Aug", y: 96 },
  { x: "Sep", y: 78 }, { x: "Oct", y: 60 },
  { x: "Nov", y: 58 }, { x: "Dec", y: 56 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    title: 'Month',
    valueType: 'Category'
  },
  primaryYAxis: {
    title: 'Temperature (Celsius)',
    labelFormat: '{value}°C'
  },
  series: [
    {
      type: 'Line', name: 'India',
      dataSource: chartData, xName: 'x', yName: 'y', width: 2,
      //Marker shape as image
      marker: {
        visible: true,
        width: 10, height: 10,
        shape: 'Image',
        imageUrl: 'sun_annotation.png'
      }
    }
  ],
  title: 'Temperature flow over months'
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customization

Marker's color and border can be customized using **fill** and **border** properties.

INDEX.JS

```

var chartData = [
    { x: 'WW', y: 12, y1: 22, y2: 38.3, y3: 50 },
    { x: 'EU', y: 9.9, y1: 26, y2: 45.2, y3: 63.6 },
    { x: 'APAC', y: 4.4, y1: 9.3, y2: 18.2, y3: 20.9 },
    { x: 'LATAM', y: 6.4, y1: 28, y2: 46.7, y3: 65.1 },
    { x: 'MEA', y: 30, y1: 45.7, y2: 61.5, y3: 73 },
    { x: 'NA', y: 25.3, y1: 35.9, y2: 64, y3: 81.4 }
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        title: 'Countries',
        valueType: 'Category', interval: 1,
        labelIntersectAction : 'Rotate45'
    },
    primaryYAxis:
    {
        title: 'Penetration (%)',
        labelFormat: '{value}%',
        minimum: 0, maximum: 90
    },
    series: [
        {
            type: 'Line', name: 'December 2007',
            dataSource: chartData, xName: 'x', yName: 'y', width: 2,
            marker: {
                visible: true,
                width: 10, height: 10,
                shape: 'Diamond'
            }
        }, {
            type: 'Line', name: 'December 2008',
            dataSource: chartData, xName: 'x', yName: 'y1', width: 2,

```

```

//Marker for chart series
marker: {
    visible: true,
    width: 10, height: 10,
    shape: 'Pentagon'
}, {
    type: 'Line', name: 'December 2009',
    dataSource: chartData, xName: 'x', yName: 'y2', width: 2,
    marker: {
        visible: true,
        width: 10, height: 10,
        shape: 'Triangle',
    }
}, {
    type: 'Line', name: 'December 2010',
    dataSource: chartData, xName: 'x', yName: 'y3', width: 2,
    marker: {
        visible: true,
        width: 10, height: 10,
        shape: 'Circle'
    }
}
],
title: 'FB Penetration of Internet Audience'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

Customizing specific point

You can also customize the specific marker and label using [pointRender](#) event. The `pointRender` event allows you to change the shape, color and border for a point.

INDEX.JS

```
var chartData = [
  { x: 'Jan', y: -7.1 }, { x: 'Feb', y: -3.7 },
  { x: 'Mar', y: 2 }, { x: 'Apr', y: 6.3 },
  { x: 'May', y: 13.3 }, { x: 'Jun', y: 18.0 },
  { x: 'Jul', y: 19.8 }, { x: 'Aug', y: 18.1 },
  { x: 'Sep', y: 13.1 }, { x: 'Oct', y: 4.1 },
  { x: 'Nov', y: -3.8 }, { x: 'Dec', y: -6.8 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    title: 'Months',
    valueType: 'Category'
  },
  primaryYAxis: {
    title: 'Temperature (Celsius)',
    minimum: -20, maximum: 25, interval: 5,
    edgeLabelPlacement: 'Shift',
    labelFormat: '{value}°C'
  },
  series: [
    {
      type: 'Line', name: 'Warmest', width: 2,
      dataSource: chartData, xName: 'x', yName: 'y',
      marker: {
        visible: true,
        height: 10, width: 10,
        shape: 'Pentagon',
        dataLabel: { visible: true }
      }
    }
  ],
  title: 'Alaska Weather Statistics - 2016',
  // pointRender event for chart
  pointRender: (args) => {
    if (args.point.index === 6) {
      args.fill = 'red'
    }
  },
  textRender: (args) => {
    if (args.point.index === 6) {
      args.text = 'Maximum Temperature';
      args.color = 'red';
    }
  }
}, '#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Fill marker with series color

Marker can be filled with the series color by setting the [isFilled](#) property to **true**.

INDEX.JS

```

var chartData = [
  { x: 'WW', y: 12, y1: 22, y2: 38.3, y3: 50 },
  { x: 'EU', y: 9.9, y1: 26, y2: 45.2, y3: 63.6 },
  { x: 'APAC', y: 4.4, y1: 9.3, y2: 18.2, y3: 20.9 },
  { x: 'LATAM', y: 6.4, y1: 28, y2: 46.7, y3: 65.1 },
  { x: 'MEA', y: 30, y1: 45.7, y2: 61.5, y3: 73 },
  { x: 'NA', y: 25.3, y1: 35.9, y2: 64, y3: 81.4 },
];
var chart = new ej.charts.Chart(
  {
    primaryXAxis: {
      title: 'Countries',
      valueType: 'Category',
      interval: 1,
      labelIntersectAction: 'Rotate45',
    },
    primaryYAxis: {
      title: 'Penetration (%)',
      labelFormat: '{value}%',
      minimum: 0,
      maximum: 90,
    },
    series: [

```

```

{
  type: 'Line',
  name: 'December 2007',
  dataSource: chartData,
  xName: 'x',
  yName: 'y',
  width: 2,
  marker: {
    visible: true,
    width: 10,
    height: 10,
    shape: 'Diamond',
    isFilled: true,
  },
},
{
  type: 'Line',
  name: 'December 2008',
  dataSource: chartData,
  xName: 'x',
  yName: 'y1',
  width: 2,
  //Marker for chart series
  marker: {
    visible: true,
    width: 10,
    height: 10,
    shape: 'Pentagon',
    isFilled: true,
  },
},
{
  type: 'Line',
  name: 'December 2009',
  dataSource: chartData,
  xName: 'x',
  yName: 'y2',
  width: 2,
  marker: {
    visible: true,
    width: 10,
    height: 10,
    shape: 'Triangle',
    isFilled: true,
  },
},
{
  type: 'Line',
  name: 'December 2010',
  dataSource: chartData,
  xName: 'x',
  yName: 'y3',
  width: 2,
  marker: {
    visible: true,
    width: 10,
    height: 10,
  },
}

```

```

        shape: 'Circle',
        isFilled: true,
    },
},
],
title: 'FB Penetration of Internet Audience',
},
'#element'
);

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

See also

- [Customize the marker with different shape](#)

Data labels in ##Platform_Name## Chart control

Data label can be added to a chart series by enabling the [visible](#) option in the dataLabel. By default, the labels will arrange smartly without overlapping.

INDEX.JS

```

var chartData = [
  { x: new Date(2016, 0, 1), y: -7.1 }, { x: new Date(2016, 1, 1), y: -3.7
},

```



```

    { x: new Date(2016, 2, 1), y: 0.8 }, { x: new Date(2016, 3, 1), y: 6.3
  },
  { x: new Date(2016, 4, 1), y: 13.3 }, { x: new Date(2016, 5, 1), y: 18.0
  },
  { x: new Date(2016, 6, 1), y: 19.8 }, { x: new Date(2016, 7, 1), y: 18.1
  },
  { x: new Date(2016, 8, 1), y: 13.1 }, { x: new Date(2016, 9, 1), y: 4.1
  },
  { x: new Date(2016, 10, 1), y: -3.8 }, { x: new Date(2016, 11, 1), y: -
6.8 }
  ];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    title: 'Months',
    valueType: 'DateTime', labelFormat: 'yMMM',
    edgeLabelPlacement: 'Shift'
  },
  primaryYAxis:
  {
    title: 'Temperature (Celsius)',
    minimum: -20, maximum: 20, interval: 10,
    edgeLabelPlacement: 'Shift',
    labelFormat: '{value}°C'
  },
  series: [
    {
      type: 'Line', name: 'Warmest', width: 2,
      dataSource: chartData, xName: 'x', yName: 'y',
      marker: {
        visible: true,
        height: 10, width: 10,
        shape: 'Pentagon',
        //Data label for chart series
        dataLabel: { visible: true }
      }
    }
  ],
  title: 'Alaska Weather Statistics - 2016'
}, '#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
```

```
<body>

  <div id="container">
    <div id="element"></div>
  </div>
</body>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Note: To use data label feature, we need to inject `DataLabel` using `Chart.Inject(DataLabel)` method.

Position

Using [position](#) property, you can place the label either on `Top`, `Middle`, `Bottom` or `Outer` (outer is applicable for column and bar type series).

INDEX.JS

```
var chartData = [
  { x: 'Jan', y: -7.1 }, { x: 'Feb', y: -3.7 },
  { x: 'Mar', y: 2 }, { x: 'Apr', y: 6.3 },
  { x: 'May', y: 13.3 }, { x: 'Jun', y: 18.0 },
  { x: 'Jul', y: 19.8 }, { x: 'Aug', y: 18.1 },
  { x: 'Sep', y: 13.1 }, { x: 'Oct', y: 4.1 },
  { x: 'Nov', y: -3.8 }, { x: 'Dec', y: -6.8 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    title: 'Months',
    valueType: 'Category'
  },
  primaryYAxis: {
    title: 'Temperature (Celsius)',
    minimum: -20, maximum: 25, interval: 5,
    edgeLabelPlacement: 'Shift',
    labelFormat: '{value}°C'
  },
  series: [
    {
      type: 'Column', name: 'Warmest',
      dataSource: chartData, xName: 'x', yName: 'y',
      marker: {
        visible: true,
        height: 10, width: 10,
        shape: 'Pentagon',
        //Data label position as middle
        dataLabel: { visible: true, position: 'Middle' }
      }
    }
  ],
  title: 'Alaska Weather Statistics - 2016'
```

```
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Note: The position **Outer** is applicable for column and bar type series.

Data Label Template

Label content can be formatted by using the template option. Inside the template, you can add the placeholder text `${point.x}` and `${point.y}` to display corresponding data points x & y value. Using [template](#) property, you can set data label template in chart.

INDEX.JS

```
var chartData = [
  { x: 'Jan', y: -7.1 }, { x: 'Feb', y: -3.7 },
  { x: 'Mar', y: 2 }, { x: 'Apr', y: 6.3 },
  { x: 'May', y: 13.3 }, { x: 'Jun', y: 18.0 },
  { x: 'Jul', y: 19.8 }, { x: 'Aug', y: 18.1 },
  { x: 'Sep', y: 13.1 }, { x: 'Oct', y: 4.1 },
  { x: 'Nov', y: -3.8 }, { x: 'Dec', y: -6.8 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    title: 'Months',
    valueType: 'Category'
  },
  primaryYAxis:
```

```

    {
        title: 'Temperature (Celsius)',
        minimum: -20, maximum: 25, interval: 5,
        edgeLabelPlacement: 'Shift',
        labelFormat: '{value}°C'
    },
    series: [
        {
            type: 'Column', name: 'Warmest',
            dataSource: chartData, xName: 'x', yName: 'y',
            marker: {
                visible: true,
                height: 10, width: 10,
                shape: 'Pentagon',
                //Data label position as middle
                dataLabel: { visible: true, position: 'Middle', margin:{
left:5, right:5, top:5, bottom:5 } }
            }
        },
    ],
    title: 'Alaska Weather Statistics - 2016'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Text Mapping

Text from the data source can be mapped using `name` property.

INDEX.JS

```

var chartData = [
  { x: 'Jan', y: -7.1 }, { x: 'Feb', y: -3.7 },
  { x: 'Mar', y: 2 }, { x: 'Apr', y: 6.3 },
  { x: 'May', y: 13.3 }, { x: 'Jun', y: 18.0 },
  { x: 'Jul', y: 19.8 }, { x: 'Aug', y: 18.1 },
  { x: 'Sep', y: 13.1 }, { x: 'Oct', y: 4.1 },
  { x: 'Nov', y: -3.8 }, { x: 'Dec', y: -6.8 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    title: 'Months',
    valueType: 'Category'
  },
  primaryYAxis: {
    title: 'Temperature (Celsius)',
    minimum: -20, maximum: 25, interval: 5,
    edgeLabelPlacement: 'Shift',
    labelFormat: '{value}°C'
  },
  series: [
    {
      type: 'Column', name: 'Warmest',
      dataSource: chartData, xName: 'x', yName: 'y',
      marker: {
        visible: true,
        height: 10, width: 10,
        shape: 'Pentagon',
        //Data label position as middle
        dataLabel: { visible: true, position: 'Middle' }
      }
    }
  ],
  title: 'Alaska Weather Statistics - 2016'
}, '#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>
```

```

<div id="container">
  <div id="element"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Format

Data label for the chart can be formatted using [format](#) property. You can use the global formatting options, such as 'n', 'p', and 'c'.

INDEX.JS

```

var chartData = [
  { x: new Date(2016, 0, 1), y: -7 }, { x: new Date(2016, 1, 1), y: -3 },
  { x: new Date(2016, 2, 1), y: 8 }, { x: new Date(2016, 3, 1), y: 6 },
  { x: new Date(2016, 4, 1), y: 13 }, { x: new Date(2016, 5, 1), y: 18 },
  { x: new Date(2016, 6, 1), y: 19 }, { x: new Date(2016, 7, 1), y: 18 },
  { x: new Date(2016, 8, 1), y: 13 }, { x: new Date(2016, 9, 1), y: 4 },
  { x: new Date(2016, 10, 1), y: -3 }, { x: new Date(2016, 11, 1), y: -6 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    title: 'Months',
    valueType: 'DateTime', labelFormat: 'yMMM',
    edgeLabelPlacement: 'Shift'
  },
  primaryYAxis: {
    {
      title: 'Temperature (Celsius)',
      minimum: -20, maximum: 20, interval: 10,
      edgeLabelPlacement: 'Shift',
      labelFormat: '{value}°C'
    }
  },
  series: [
    {
      type: 'Line', name: 'Warmest', width: 2,
      dataSource: chartData, xName: 'x', yName: 'y',
      marker: {
        visible: true,
        height: 10, width: 10,
        shape: 'Pentagon', isFilled: true,
        //Data label for chart series
        dataLabel: { visible: true , format:"n2"}
      }
    }
  ],
  title: 'Alaska Weather Statistics - 2016'
}, '#element');

```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Value	Format	Resultant Value	Description
1000	n1	1000.0	The number is rounded to 1 decimal place.
1000	n2	1000.00	The number is rounded to 2 decimal places.
1000	n3	1000.000	The number is rounded to 3 decimal place.
0.01	p1	1.0%	The number is converted to percentage with 1 decimal place.
0.01	p2	1.00%	The number is converted to percentage with 2 decimal place.
0.01	p3	1.000%	The number is converted to percentage with 3 decimal place.
1000	c1	\$1000.0	The currency symbol is appended to number and number is rounded to 1 decimal place.
1000	c2	\$1000.00	The currency symbol is appended to number and number is rounded to 2 decimal place.

Margin

margin for data label can be applied to using left, right, bottom and top properties.

INDEX.JS

```

var chartData = [
  { x: 'Jan', y: -7.1 }, { x: 'Feb', y: -3.7 },
  { x: 'Mar', y: 2 }, { x: 'Apr', y: 6.3 },
  { x: 'May', y: 13.3 }, { x: 'Jun', y: 18.0 },
  { x: 'Jul', y: 19.8 }, { x: 'Aug', y: 18.1 },
  { x: 'Sep', y: 13.1 }, { x: 'Oct', y: 4.1 },
  { x: 'Nov', y: -3.8 }, { x: 'Dec', y: -6.8 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    title: 'Months',
    valueType: 'Category'
  },
  primaryYAxis: {
    title: 'Temperature (Celsius)',
    minimum: -20, maximum: 25, interval: 5,
    edgeLabelPlacement: 'Shift',
    labelFormat: '{value}°C'
  },
  series: [
    {
      type: 'Column', name: 'Warmest',
      dataSource: chartData, xName: 'x', yName: 'y',
      marker: {
        visible: true,
        height: 10, width: 10,
        shape: 'Pentagon',
        //Data label position as middle
        dataLabel: { visible: true, position: 'Middle', margin:{
left:5, right:5, top:5, bottom:5 } }
      }
    }
  ],
  title: 'Alaska Weather Statistics - 2016'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">

```



```

        <div id="element"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customization

stroke and border of data label can be customized using fill and border properties. Rounded corners can be customized using rx and ry properties.

INDEX.JS

```

var chartData = [
    { x: 'Jan', y: -7.1 }, { x: 'Feb', y: -3.7 },
    { x: 'Mar', y: 2 }, { x: 'Apr', y: 6.3 },
    { x: 'May', y: 13.3 }, { x: 'Jun', y: 18.0 },
    { x: 'Jul', y: 19.8 }, { x: 'Aug', y: 18.1 },
    { x: 'Sep', y: 13.1 }, { x: 'Oct', y: 4.1 },
    { x: 'Nov', y: -3.8 }, { x: 'Dec', y: -6.8 }
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        title: 'Months',
        valueType: 'Category'
    },
    primaryYAxis: {
        title: 'Temperature (Celsius)',
        minimum: -20, maximum: 25, interval: 5,
        edgeLabelPlacement: 'Shift',
        labelFormat: '{value}°C'
    },
    series: [
        {
            type: 'Column', name: 'Warmest',
            dataSource: chartData, xName: 'x', yName: 'y',
            marker: {
                visible: true,
                height: 10, width: 10,
                shape: 'Pentagon',
                //Data label position as middle
                dataLabel: { visible: true, position: 'Middle', margin:{
                    left:5, right:5, top:5, bottom:5 } }
            }
        }
    ],
    title: 'Alaska Weather Statistics - 2016'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Note: `rx` and `ry` properties requires `border` values not to be null.

Customizing Specific Point

You can also customize the specific marker and label using [pointRender](#) and [textRender](#) event. `pointRender` event allows you to change the shape, color and border for a point, whereas the `textRender` event allows you to change the text for the point.

INDEX.JS

```

var chartData = [
  { x: 'Jan', y: -7.1 }, { x: 'Feb', y: -3.7 },
  { x: 'Mar', y: 2 }, { x: 'Apr', y: 6.3 },
  { x: 'May', y: 13.3 }, { x: 'Jun', y: 18.0 },
  { x: 'Jul', y: 19.8 }, { x: 'Aug', y: 18.1 },
  { x: 'Sep', y: 13.1 }, { x: 'Oct', y: 4.1 },
  { x: 'Nov', y: -3.8 }, { x: 'Dec', y: -6.8 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    title: 'Months',
    valueType: 'Category'
  },
  primaryYAxis: {
    title: 'Temperature (Celsius)',
    minimum: -20, maximum: 25, interval: 5,
    edgeLabelPlacement: 'Shift',
    labelFormat: '{value}°C'
  }
});

```

```

    },
    series: [
        {
            type: 'Line', name: 'Warmest', width: 2,
            dataSource: chartData, xName: 'x', yName: 'y',
            marker: {
                visible: true,
                height: 10, width: 10,
                shape: 'Pentagon',
                dataLabel: { visible: true }
            }
        }
    ],
    title: 'Alaska Weather Statistics - 2016',
    // pointRender event for chart
    pointRender: (args) => {
        if (args.point.index === 6) {
            args.fill = 'red'
        }
    },
    textRender: (args) => {
        if (args.point.index === 6) {
            args.text = 'Maximum Temperature';
            args.color = 'red';
        }
    }
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
<script src="index.js" type="text/javascript"></script>

```

```
</body></html>
```

Show percentage based on each series points

You can calculate the percentage value based on the sum for each series using the `seriesRender` and `textRender` events in the chart. In `seriesRender` calculate the sum of each series y values and In `textRender` calculate percentage value based on the sum value and modify the text.

INDEX.JS

```
var total = [];
var chart = new ej.charts.Chart({
  //Initializing Primary X Axis
  primaryXAxis: {
    valueType: 'Category', interval: 1, majorGridLines: { width: 0 }
  },
  chartArea: { border: { width: 0 } },
  //Initializing Primary Y Axis
  primaryYAxis: {
    majorGridLines: { width: 0 },
    majorTickLines: { width: 0 }, lineStyle: { width: 0 }, labelStyle: {
color: 'transparent' }
  },
  //Initializing Chart Series
  series: [
    {
      type: 'Column', xName: 'x', width: 2, yName: 'y', name: 'Gold',
      dataSource: [{ x: 'USA', y: 46 }, { x: 'GBR', y: 27 }, { x:
'CHN', y: 26 }],
      marker: { dataLabel: { visible: true, position: 'Top', font: {
fontWeight: '600', color: '#ffffff' } } }
    },
    {
      type: 'Column', xName: 'x', width: 2, yName: 'y', name:
'Silver',
      dataSource: [{ x: 'USA', y: 37 }, { x: 'GBR', y: 23 }, { x:
'CHN', y: 18 }],
      marker: { dataLabel: { visible: true, position: 'Top', font: {
fontWeight: '600', color: '#ffffff' } } }
    },
    {
      type: 'Column', xName: 'x', width: 2, yName: 'y', name:
'Bronze',
      dataSource: [{ x: 'USA', y: 38 }, { x: 'GBR', y: 17 }, { x:
'CHN', y: 26 }],
      marker: { dataLabel: { visible: true, position: 'Top', font: {
fontWeight: '600', color: '#ffffff' } } }
    }
  ],
  //Initializing Chart Title
  title: 'Olympic Medal Counts - RIO', tooltip: { enable: true },
  width: ej.base.Browser.isDevice ? '100%' : '60%',
  seriesRender: function(args) {
    for (var i = 0; i < args.data.length; i++) {
      if (!total[args.data[i].x]) total[args.data[i].x] = 0;
      total[args.data[i].x] += parseInt(args.data[i].y);
    }
  }
});
```

```

    }
  },
  textRender: function(args) {
    var percentage = (parseInt(args.text) / total[args.point.x]) * 100;
    percentage = percentage % 1 === 0 ? percentage :
percentage.toFixed(2);
    args.text = percentage + '%';
  },
});
chart.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Show total stacking values in data label](#)
- [Prevent the data label when the data value is 0](#)

Chart annotations in ##Platform_Name## Chart control

Annotations are used to mark the specific area of interest in the chart area with texts, shapes or images.

<!-- markdownlint-disable MD033 -->

You can add annotations to the chart by using the `annotations` option. By using the [content](#) option of annotation object, you can specify either the id of the element or directly specify the element in the content that needs to be displayed in the chart area.

INDEX.JS

```

var chartData = [
    { country: "USA", gold: 50, silver: 70, bronze: 45 },
    { country: "China", gold: 40, silver: 60, bronze: 55 },
    { country: "Japan", gold: 70, silver: 60, bronze: 50 },
    { country: "Australia", gold: 60, silver: 56, bronze: 40 },
    { country: "France", gold: 50, silver: 45, bronze: 35 },
    { country: "Germany", gold: 40, silver: 30, bronze: 22 },
    { country: "Italy", gold: 40, silver: 35, bronze: 37 },
    { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        valueType: 'Category',
        // label placement as on ticks
        labelPlacement: 'OnTicks',
        title: 'Countries'
    },
    primaryYAxis: {
        minimum: 0, maximum: 80,
        interval: 20, title: 'Medals'
    },
    annotations:[{
        content: '70 Gold Medals',
        region: 'Series',
        coordinateUnits: 'Point',
        x: 'Japan',
        y: 75
    }],
    series:[{
        dataSource: chartData,
        xName: 'country', yName: 'gold',
        name: 'Gold', type: 'Column'
    }, {
        dataSource: chartData,
        xName: 'country', yName: 'silver',
        name: 'Silver', type: 'Column'
    }, {
        dataSource: chartData,
        xName: 'country', yName: 'bronze',
        name: 'Bronze', type: 'Column'
    }],
    title: 'Olympic Medals'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
        <div id="element1"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Note: To use annotations feature, we need to inject **ChartAnnotation** using **Chart.Inject(ChartAnnotation)** method.

Region

Annotations can be placed either with respect to **Series** or **Chart**. by default, it will placed with respect to **Chart**.

INDEX.JS

```

var chartData = [
    { country: "USA", gold: 50, silver: 70, bronze: 45 },
    { country: "China", gold: 40, silver: 60, bronze: 55 },
    { country: "Japan", gold: 70, silver: 60, bronze: 50 },
    { country: "Australia", gold: 60, silver: 56, bronze: 40 },
    { country: "France", gold: 50, silver: 45, bronze: 35 },
    { country: "Germany", gold: 40, silver: 30, bronze: 22 },
    { country: "Italy", gold: 40, silver: 35, bronze: 37 },
    { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        valueType: 'Category',
        // label placement as on ticks
        labelPlacement: 'OnTicks',
        title: 'Countries'
    },
    primaryYAxis: {
        minimum: 0, maximum: 80,
        interval: 20, title: 'Medals'
    },
    annotations:[{
        content: '70 Gold Medals',
        region: 'Series',
        coordinateUnits: 'Point',

```

```

        x: 'Japan',
        y: 75
    }],
    series:[{
        dataSource: chartData,
        xName: 'country', yName: 'gold',
        name: 'Gold', type: 'Column'
    }, {
        dataSource: chartData,
        xName: 'country', yName: 'silver',
        name: 'Silver', type: 'Column'
    }, {
        dataSource: chartData,
        xName: 'country', yName: 'bronze',
        name: 'Bronze', type: 'Column'
    }],
    title: 'Olympic Medals'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
    <div id="element1"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Co-ordinate Units

Specified the coordinates units of the annotation either **Pixel** or **Point**.

INDEX.JS


```

var chartData = [
    { country: "USA", gold: 50, silver: 70, bronze: 45 },
    { country: "China", gold: 40, silver: 60, bronze: 55 },
    { country: "Japan", gold: 70, silver: 60, bronze: 50 },
    { country: "Australia", gold: 60, silver: 56, bronze: 40 },
    { country: "France", gold: 50, silver: 45, bronze: 35 },
    { country: "Germany", gold: 40, silver: 30, bronze: 22 },
    { country: "Italy", gold: 40, silver: 35, bronze: 37 },
    { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        valueType: 'Category',
        // label placement as on ticks
        labelPlacement: 'OnTicks',
        title: 'Countries'
    },
    primaryYAxis: {
        minimum: 0, maximum: 80,
        interval: 20, title: 'Medals'
    },
    annotations:[{
        content: '70 Gold Medals',
        region: 'Series',
        coordinateUnits: 'Point',
        x: 'Japan',
        y: 75
    }],
    series:[{
        dataSource: chartData,
        xName: 'country', yName: 'gold',
        name: 'Gold', type: 'Column'
    }, {
        dataSource: chartData,
        xName: 'country', yName: 'silver',
        name: 'Silver', type: 'Column'
    }, {
        dataSource: chartData,
        xName: 'country', yName: 'bronze',
        name: 'Bronze', type: 'Column'
    }],
    title: 'Olympic Medals'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
        <div id="element1"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Alignment

Annotation provides `verticalAlignment` and `horizontalAlignment`. The `verticalAlignment` can be customized via `Top`, `Bottom` or `Middle` and the `horizontalAlignment` can be customized via `Near`, `Far` or `Center`.

INDEX.JS

```

var chartData = [
    { country: "USA", gold: 50, silver: 70, bronze: 45 },
    { country: "China", gold: 40, silver: 60, bronze: 55 },
    { country: "Japan", gold: 70, silver: 60, bronze: 50 },
    { country: "Australia", gold: 60, silver: 56, bronze: 40 },
    { country: "France", gold: 50, silver: 45, bronze: 35 },
    { country: "Germany", gold: 40, silver: 30, bronze: 22 },
    { country: "Italy", gold: 40, silver: 35, bronze: 37 },
    { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        valueType: 'Category',
        // label placement as on ticks
        labelPlacement: 'OnTicks',
        title: 'Countries'
    },
    primaryYAxis: {
        minimum: 0, maximum: 80,
        interval: 20, title: 'Medals'
    },
    annotations:[{
        content: '70 Gold Medals',
        region: 'Series',
        coordinateUnits: 'Point',
        x: 'Japan',
        y: 75
    }],
});

```

```

series:[{
  dataSource: chartData,
  xName: 'country', yName: 'gold',
  name: 'Gold', type: 'Column'
}, {
  dataSource: chartData,
  xName: 'country', yName: 'silver',
  name: 'Silver', type: 'Column'
}, {
  dataSource: chartData,
  xName: 'country', yName: 'bronze',
  name: 'Bronze', type: 'Column'
}],
title: 'Olympic Medals'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
    <div id="element1"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Adding y-axis sub title through on annotation

By setting text div in the **content** option of annotation object you can add sub title to chart y-axis. Specified the **coordinate** value as **pixel** and customize x and y location of the text.

INDEX.JS

```

var columnData = [{ country: "USA", gold: 50 }, { country: "China", gold: 40 },
{ country: "Japan", gold: 70 },

```

```

{ country: "Australia", gold: 60 }, { country: "France", gold: 50 }, {
country: "Germany", gold: 40 },
{ country: "Italy", gold: 40 }, { country: "Sweden", gold: 30 }];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    valueType: 'Category',
  },
  primaryYAxis: {
    title: '(m2/min)'
  },
  annotations: [{
    content: '<div id="text" style="transform: rotate(-90deg);">Speed
Rate</div>',
    x: 6,
    y: 180,
    coordinateUnits: 'Pixel',
    Region: 'Chart'
  }],
  series: [{
    dataSource: columnData,
    xName: 'country', yName: 'gold',
    type: 'Column'
  }],
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
    <div id="element1"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Show total stacking values in data label](#)
- [Create footer and watermark for chart](#)

<!-- markdownlint-disable MD036 -->

Legend in ##Platform_Name## Chart control

<!-- markdownlint-disable MD036 -->

Legend provides information about the series rendered in the chart.

Position and Alignment

By using the [position](#) property, you can position the legend at left, right, top or bottom of the chart. The legend is positioned at the bottom of the chart, by default.

INDEX.JS

```
var chartData = [
  { country: "USA", gold: 50, silver: 70, bronze: 45 },
  { country: "China", gold: 40, silver: 60, bronze: 55 },
  { country: "Japan", gold: 70, silver: 60, bronze: 50 },
  { country: "Australia", gold: 60, silver: 56, bronze: 40 },
  { country: "France", gold: 50, silver: 45, bronze: 35 },
  { country: "Germany", gold: 40, silver: 30, bronze: 22 },
  { country: "Italy", gold: 40, silver: 35, bronze: 37 },
  { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    valueType: 'Category',
    title: 'Countries'
  },
  primaryYAxis: {
    minimum: 0, maximum: 80,
    interval: 20, title: 'Medals'
  },
  series:[{
    dataSource: chartData,
    xName: 'country', yName: 'gold',
    name: 'Gold', type: 'Column'
  }, {
    dataSource: chartData,
    xName: 'country', yName: 'silver',
    name: 'Silver', type: 'Column'
  }, {
    dataSource: chartData,
    xName: 'country', yName: 'bronze',
    name: 'Bronze', type: 'Column'
  }],
  title: 'Olympic Medals',
  legendSettings: {
    visible: true,
    //Legend position as top
    position:'Top'
  }
});
```

```
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Custom position helps you to position the legend anywhere in the chart using x, y coordinates.

INDEX.JS

```
var chartData = [
  { country: "USA", gold: 50, silver: 70, bronze: 45 },
  { country: "China", gold: 40, silver: 60, bronze: 55 },
  { country: "Japan", gold: 70, silver: 60, bronze: 50 },
  { country: "Australia", gold: 60, silver: 56, bronze: 40 },
  { country: "France", gold: 50, silver: 45, bronze: 35 },
  { country: "Germany", gold: 40, silver: 30, bronze: 22 },
  { country: "Italy", gold: 40, silver: 35, bronze: 37 },
  { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    valueType: 'Category'
  },
  series:[{
    dataSource: chartData,
    xName: 'country', yName: 'gold',
    name: 'Gold', type: 'Column'
  }, {
    dataSource: chartData,
```

```

        xName: 'country', yName: 'silver',
        name: 'Silver', type: 'Column'
    }, {
        dataSource: chartData,
        xName: 'country', yName: 'bronze',
        name: 'Bronze', type: 'Column'
    }],
    legendSettings: {
        visible: true,
        //Legend position as custom
        position: 'Custom',
        location: { x: 200, y: 20 } }
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Legend Reverse

You can reverse the order of the legend items by using the [reverse](#) property. By default, legend for the first series in the collection will be placed first.

INDEX.JS

```

var chartData = [
  { country: "USA", gold: 50, silver: 70, bronze: 45 },
  { country: "China", gold: 40, silver: 60, bronze: 55 },
  { country: "Japan", gold: 70, silver: 60, bronze: 50 },
  { country: "Australia", gold: 60, silver: 56, bronze: 40 },

```

```

        { country: "France", gold: 50, silver: 45, bronze: 35 },
        { country: "Germany", gold: 40, silver: 30, bronze: 22 },
        { country: "Italy", gold: 40, silver: 35, bronze: 37 },
        { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
    ];
    var chart = new ej.charts.Chart({
        primaryXAxis: {
            valueType: 'Category',
            title: 'Countries'
        },
        primaryYAxis: {
            minimum: 0, maximum: 80,
            interval: 20, title: 'Medals'
        },
        series:[{
            dataSource: chartData,
            xName: 'country', yName: 'gold',
            name: 'Gold', type: 'Column'
        }, {
            dataSource: chartData,
            xName: 'country', yName: 'silver',
            name: 'Silver', type: 'Column'
        }, {
            dataSource: chartData,
            xName: 'country', yName: 'bronze',
            name: 'Bronze', type: 'Column'
        }],
        title: 'Olympic Medals',
        legendSettings: {
            visible: true,
            reverse: true
        }
    }, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
</body>
</html>

```



```

var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Legend Alignment

You can align the legend as center, far or near to the chart using [alignment](#) property.

INDEX.JS

```

var chartData = [
    { country: "USA", gold: 50, silver: 70, bronze: 45 },
    { country: "China", gold: 40, silver: 60, bronze: 55 },
    { country: "Japan", gold: 70, silver: 60, bronze: 50 },
    { country: "Australia", gold: 60, silver: 56, bronze: 40 },
    { country: "France", gold: 50, silver: 45, bronze: 35 },
    { country: "Germany", gold: 40, silver: 30, bronze: 22 },
    { country: "Italy", gold: 40, silver: 35, bronze: 37 },
    { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        valueType: 'Category',
        title: 'Countries'
    },
    primaryYAxis: {
        minimum: 0, maximum: 80,
        interval: 20, title: 'Medals'
    },
    series:[{
        dataSource: chartData,
        xName: 'country', yName: 'gold',
        name: 'Gold', type: 'Column'
    }, {
        dataSource: chartData,
        xName: 'country', yName: 'silver',
        name: 'Silver', type: 'Column'
    }, {
        dataSource: chartData,
        xName: 'country', yName: 'bronze',
        name: 'Bronze', type: 'Column'
    }],
    title: 'Olympic Medals',
    legendSettings: {
        visible: true,
        position: 'Top',
        //Legend alignment as near
        alignment: 'Near' }
    }, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Customization

To change the legend icon shape, you can use [legendShape](#) property in the [series](#). By default legend icon shape is `seriesType`.

INDEX.JS

```

var chartData = [
  { country: "USA", gold: 50, silver: 70, bronze: 45 },
  { country: "China", gold: 40, silver: 60, bronze: 55 },
  { country: "Japan", gold: 70, silver: 60, bronze: 50 },
  { country: "Australia", gold: 60, silver: 56, bronze: 40 },
  { country: "France", gold: 50, silver: 45, bronze: 35 },
  { country: "Germany", gold: 40, silver: 30, bronze: 22 },
  { country: "Italy", gold: 40, silver: 35, bronze: 37 },
  { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    valueType: 'Category',
    title: 'Countries'
  },
  primaryYAxis: {
    minimum: 0, maximum: 80,
    interval: 20, title: 'Medals'
  },
  series:[{
    dataSource: chartData,
    xName: 'country', yName: 'gold',

```

```

        name: 'Gold', type: 'Column',
        //Legend icon type for chart
        legendShape: 'Circle'
    }, {
        dataSource: chartData,
        xName: 'country', yName: 'silver',
        name: 'Silver', type: 'Column',
        legendShape: 'SeriesType'
    }, {
        dataSource: chartData,
        xName: 'country', yName: 'bronze',
        name: 'Bronze', type: 'Column',
        legendShape: 'Rectangle'
    }],
    title: 'Olympic Medals',
    legendSettings: { visible: true }
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Legend Size

By default, legend takes 20% - 25% of the chart's height horizontally, when it is placed on top or bottom position and 20% - 25% of the width vertically, while placing on left or right position of the chart. You can change this default legend size by using the [width](#) and [height](#) property of the `legendSettings`.

INDEX.JS

```

var chartData = [
    { country: "USA", gold: 50, silver: 70, bronze: 45 },
    { country: "China", gold: 40, silver: 60, bronze: 55 },
    { country: "Japan", gold: 70, silver: 60, bronze: 50 },
    { country: "Australia", gold: 60, silver: 56, bronze: 40 },
    { country: "France", gold: 50, silver: 45, bronze: 35 },
    { country: "Germany", gold: 40, silver: 30, bronze: 22 },
    { country: "Italy", gold: 40, silver: 35, bronze: 37 },
    { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        valueType: 'Category',
        title: 'Countries'
    },
    primaryYAxis: {
        minimum: 0, maximum: 80,
        interval: 20, title: 'Medals'
    },
    series:[{
        dataSource: chartData,
        xName: 'country', yName: 'gold',
        name: 'Gold', type: 'Column', legendShape: 'Circle'
    }, {
        dataSource: chartData,
        xName: 'country', yName: 'silver',
        name: 'Silver', type: 'Column', legendShape: 'Circle'
    }, {
        dataSource: chartData,
        xName: 'country', yName: 'bronze',
        name: 'Bronze', type: 'Column', legendShape: 'Circle'
    }],
    title: 'Olympic Medals',
    legendSettings: {
        visible: true,
        //Legend size for chart
        width: '500', height: '100',
        border: { width: 1, color: 'pink' }
    }
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>

```

```

</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Legend Item Size

You can customize the size of the legend items by using the [shapeHeight](#) and [shapeWidth](#) property.

INDEX.JS

```

var chartData = [
    { country: "USA", gold: 50, silver: 70, bronze: 45 },
    { country: "China", gold: 40, silver: 60, bronze: 55 },
    { country: "Japan", gold: 70, silver: 60, bronze: 50 },
    { country: "Australia", gold: 60, silver: 56, bronze: 40 },
    { country: "France", gold: 50, silver: 45, bronze: 35 },
    { country: "Germany", gold: 40, silver: 30, bronze: 22 },
    { country: "Italy", gold: 40, silver: 35, bronze: 37 },
    { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        valueType: 'Category',
        title: 'Countries'
    },
    primaryYAxis: {
        minimum: 0, maximum: 80,
        interval: 20, title: 'Medals'
    },
    series:[{
        dataSource: chartData,
        xName: 'country', yName: 'gold',
        name: 'Gold', type: 'Column', legendShape: 'Circle'
    }, {
        dataSource: chartData,
        xName: 'country', yName: 'silver',
        name: 'Silver', type: 'Column', legendShape: 'Circle'
    }, {
        dataSource: chartData,
        xName: 'country', yName: 'bronze',
        name: 'Bronze', type: 'Column', legendShape: 'Circle'
    }],
    title: 'Olympic Medals',
    legendSettings: {
        visible: true,
        //Legend item size for chart
    }
});

```

```

        shapeHeight: 10, shapeWidth: 10
    }
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Paging for Legend

Paging will be enabled by default, when the legend items exceeds the legend bounds. You can view each legend items by navigating between the pages using navigation buttons.

INDEX.JS

```

var chartData = [
  { x: 'WW', y: 12, y1: 22, y2: 38.3, y3: 50 },
  { x: 'EU', y: 9.9, y1: 26, y2: 45.2, y3: 63.6 },
  { x: 'APAC', y: 4.4, y1: 9.3, y2: 18.2, y3: 20.9 },
  { x: 'LATAM', y: 6.4, y1: 28, y2: 46.7, y3: 65.1 },
  { x: 'MEA', y: 30, y1: 45.7, y2: 61.5, y3: 73 },
  { x: 'NA', y: 25.3, y1: 35.9, y2: 64, y3: 81.4 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    title: 'Countries',
    valueType: 'Category', interval: 1,
    labelIntersectAction : 'Rotate45'
  },
  primaryYAxis:

```

```

{
    title: 'Penetration (%)',
    labelFormat: '{value}%',
    minimum: 0, maximum: 90
},
series: [
    {
        type: 'Line', name: 'December 2007',
        dataSource: chartData, xName: 'x', yName: 'y', width: 2,
        marker: {
            visible: true,
            width: 10, height: 10,
            shape: 'Diamond'
        }
    }, {
        type: 'Line', name: 'December 2008',
        dataSource: chartData, xName: 'x', yName: 'y1', width: 2,
        marker: {
            visible: true,
            width: 10, height: 10,
            shape: 'Pentagon'
        }
    }, {
        type: 'Line', name: 'December 2009',
        dataSource: chartData, xName: 'x', yName: 'y2', width: 2,
        marker: {
            visible: true,
            width: 10, height: 10,
            shape: 'Triangle'
        }
    }, {
        type: 'Line', name: 'December 2010',
        dataSource: chartData, xName: 'x', yName: 'y3', width: 2,
        marker: {
            visible: true,
            width: 10, height: 10,
            shape: 'Circle'
        }
    }
],
title: 'FB Penetration of Internet Audience',
legendSettings: {
    padding: 10, shapePadding: 10,
    visible: true, border: {
        width: 2, color: 'grey'
    },
    width: '200'
}
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

```

```

<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Legend Text Wrap

When the legend text exceeds the container, the text can be wrapped by using [textWrap](#) Property. End user can also wrap the legend text based on the [maximumLabelWidth](#) property.

INDEX.JS

```

var chartData = [
    { country: "USA", gold: 50, silver: 70, bronze: 45 },
    { country: "China", gold: 40, silver: 60, bronze: 55 },
    { country: "Japan", gold: 70, silver: 60, bronze: 50 },
    { country: "Australia", gold: 60, silver: 56, bronze: 40 },
    { country: "France", gold: 50, silver: 45, bronze: 35 },
    { country: "Germany", gold: 40, silver: 30, bronze: 22 },
    { country: "Italy", gold: 40, silver: 35, bronze: 37 },
    { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        valueType: 'Category',
        title: 'Countries'
    },
    primaryYAxis: {
        minimum: 0, maximum: 80,
        interval: 20, title: 'Medals'
    },
    series:[{
        dataSource: chartData,
        xName: 'country', yName: 'gold',
        name: 'Gold Medals', type: 'Column'
    }, {
        dataSource: chartData,
        xName: 'country', yName: 'silver',

```



```

        name: 'Silver Medals', type: 'Column'
    }, {
        dataSource: chartData,
        xName: 'country', yName: 'bronze',
        name: 'Bronze Medals', type: 'Column'
    }],
    title: 'Olympic Medals',
    legendSettings: {
        visible: true,
        position: 'Right',
        textWrap: 'Wrap',
        maximumLabelWidth: 50,
    }
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Set the label color based on series color

You can set the legend label color based on series color by using chart's [loaded](#) event.

INDEX.JS

```

var chartData = [
  { country: "USA", gold: 50, silver: 70, bronze: 45 },
  { country: "China", gold: 40, silver: 60, bronze: 55 },
  { country: "Japan", gold: 70, silver: 60, bronze: 50 },
  { country: "Australia", gold: 60, silver: 56, bronze: 40 },
  { country: "France", gold: 50, silver: 45, bronze: 35 },

```

```

    { country: "Germany", gold: 40, silver: 30, bronze: 22 },
    { country: "Italy", gold: 40, silver: 35, bronze: 37 },
    { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
  ];
  // declare the series colors
  var colors = ['#00BDAE', '#404041', '#357CD2'];
  var chart = new ej.charts.Chart({
    primaryXAxis: {
      valueType: 'Category',
      title: 'Countries'
    },
    primaryYAxis: {
      minimum: 0, maximum: 80,
      interval: 20, title: 'Medals'
    },
    series:[{
      dataSource: chartData,
      xName: 'country', yName: 'gold',
      name: 'Gold', type: 'Column'
    }, {
      dataSource: chartData,
      xName: 'country', yName: 'silver',
      name: 'Silver', type: 'Column'
    }, {
      dataSource: chartData,
      xName: 'country', yName: 'bronze',
      name: 'Bronze', type: 'Column'
    }],
    title: 'Olympic Medals',
    loaded: (args) => {
      let chart = document.querySelector('.e-chart');
      let legendTextCol =
chart.querySelectorAll('[id*="chart_legend_text_"]');
      for (let i = 0; i < legendTextCol.length; i++) {
        //set the color to legend label
        legendTextCol[i].setAttribute('fill', colors[i]);
      }
    },
  }, '#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
```

```
<body>

  <div id="container">
    <div id="element"></div>
  </div>
</body>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Series selection on Legend

By default, legend click enables you to collapse the series visibility. On other hand, if you need to select a series through legend click, disable the [toggleVisibility](#).

INDEX.JS

```
var chartData = [
  { country: "USA", gold: 50, silver: 70, bronze: 45 },
  { country: "China", gold: 40, silver: 60, bronze: 55 },
  { country: "Japan", gold: 70, silver: 60, bronze: 50 },
  { country: "Australia", gold: 60, silver: 56, bronze: 40 },
  { country: "France", gold: 50, silver: 45, bronze: 35 },
  { country: "Germany", gold: 40, silver: 30, bronze: 22 },
  { country: "Italy", gold: 40, silver: 35, bronze: 37 },
  { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
// declare the series colors
var colors = ['#00BDAE', '#404041', '#357CD2'];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    valueType: 'Category',
    title: 'Countries'
  },
  primaryYAxis: {
    minimum: 0, maximum: 80,
    interval: 20, title: 'Medals'
  },
  series:[{
    dataSource: chartData,
    xName: 'country', yName: 'gold',
    name: 'Gold', type: 'Column'
  }, {
    dataSource: chartData,
    xName: 'country', yName: 'silver',
    name: 'Silver', type: 'Column'
  }, {
    dataSource: chartData,
    xName: 'country', yName: 'bronze',
    name: 'Bronze', type: 'Column'
  }],
  title: 'Olympic Medals',
  loaded: (args) => {
```

```

    let chart = document.querySelector('.e-chart');;
    let legendTextCol =
chart.querySelectorAll('[id*="chart_legend_text_"]');
    for (let i = 0; i < legendTextCol.length; i++) {
        //set the color to legend label
        legendTextCol[i].setAttribute('fill', colors[i]);
    }
},
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Collapsing Legend Item

By default, series name will be displayed as legend. To skip the legend for a particular series, you can give empty string to the series name.

INDEX.JS

```

var chartData = [
  { country: "USA", gold: 50, silver: 70, bronze: 45 },
  { country: "China", gold: 40, silver: 60, bronze: 55 },
  { country: "Japan", gold: 70, silver: 60, bronze: 50 },
  { country: "Australia", gold: 60, silver: 56, bronze: 40 },
  { country: "France", gold: 50, silver: 45, bronze: 35 },
  { country: "Germany", gold: 40, silver: 30, bronze: 22 },
  { country: "Italy", gold: 40, silver: 35, bronze: 37 },
  { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
]

```

```

];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    valueType: 'Category',
    title: 'Countries'
  },
  primaryYAxis: {
    minimum: 0, maximum: 80,
    interval: 20, title: 'Medals'
  },
  series:[{
    dataSource: chartData,
    xName: 'country', yName: 'gold',
    name: 'Gold', type: 'Column', legendShape: 'Circle'
  }, {
    dataSource: chartData,
    xName: 'country', yName: 'silver',
    type: 'Column', legendShape: 'Circle'
  }, {
    dataSource: chartData,
    xName: 'country', yName: 'bronze',
    name: 'Bronze', type: 'Column', legendShape: 'Circle'
  }],
  title: 'Olympic Medals',
  legendSettings: {
    visible: true,
    toggleSeriesVisibility: true
  }
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}

```

```

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Legend Title

You can set title for legend using `title` property in `legendSettings`. You can also customize the `fontStyle`, `size`, `fontWeight`,

`color`, `textAlignment`, `fontFamily`, `opacity` and `textOverflow` of legend title. `titlePosition` is used to set the legend position in `Top`, `Left` and `Right` position. `maximumTitleWidth` is used to set the width of the legend title. By default, it will be `100px`.

INDEX.JS

```

var chartData = [
  { country: "USA", gold: 50, silver: 70, bronze: 45 },
  { country: "China", gold: 40, silver: 60, bronze: 55 },
  { country: "Japan", gold: 70, silver: 60, bronze: 50 },
  { country: "Australia", gold: 60, silver: 56, bronze: 40 },
  { country: "France", gold: 50, silver: 45, bronze: 35 },
  { country: "Germany", gold: 40, silver: 30, bronze: 22 },
  { country: "Italy", gold: 40, silver: 35, bronze: 37 },
  { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    valueType: 'Category',
  },
  primaryYAxis: {
    minimum: 0, maximum: 80,
    interval: 20, title: 'Medals'
  },
  series:[{
    dataSource: chartData,
    xName: 'country', yName: 'gold',
    name: 'Gold', type: 'Column',
    //Legend icon type for chart
    legendShape: 'Circle'
  }, {
    dataSource: chartData,
    xName: 'country', yName: 'silver',
    name: 'Silver', type: 'Column',
    legendShape: 'SeriesType'
  }, {
    dataSource: chartData,
    xName: 'country', yName: 'bronze',
    name: 'Bronze', type: 'Column',
    legendShape: 'Rectangle'
  }],
  title: 'Olympic Medals',
  legendSettings: { visible: true, title: 'Countries' }
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Arrow Page Navigation

By default, the page number will be enabled while legend paging. Now, you can disable that page number and also you can get left and right arrows for page navigation. You have to set `false` value to `enablePages` to get this support.

INDEX.JS

```

var chartData = [
  { x: 'WW', y: 12, y1: 22, y2: 38.3, y3: 50 },
  { x: 'EU', y: 9.9, y1: 26, y2: 45.2, y3: 63.6 },
  { x: 'APAC', y: 4.4, y1: 9.3, y2: 18.2, y3: 20.9 },
  { x: 'LATAM', y: 6.4, y1: 28, y2: 46.7, y3: 65.1 },
  { x: 'MEA', y: 30, y1: 45.7, y2: 61.5, y3: 73 },
  { x: 'NA', y: 25.3, y1: 35.9, y2: 64, y3: 81.4 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    title: 'Countries',
    valueType: 'Category', interval: 1,
    labelIntersectAction : 'Rotate45'
  },
  primaryYAxis:
  {
    title: 'Penetration (%)',
    labelFormat: '{value}%',
    minimum: 0, maximum: 90
  },
  series: [

```

```

        {
            type: 'Line', name: 'December 2007',
            dataSource: chartData, xName: 'x', yName: 'y', width: 2,
            marker: {
                visible: true,
                width: 10, height: 10,
                shape: 'Diamond'
            }
        }, {
            type: 'Line', name: 'December 2008',
            dataSource: chartData, xName: 'x', yName: 'y1', width: 2,
            marker: {
                visible: true,
                width: 10, height: 10,
                shape: 'Pentagon'
            }
        }, {
            type: 'Line', name: 'December 2009',
            dataSource: chartData, xName: 'x', yName: 'y2', width: 2,
            marker: {
                visible: true,
                width: 10, height: 10,
                shape: 'Triangle',
            }
        }, {
            type: 'Line', name: 'December 2010',
            dataSource: chartData, xName: 'x', yName: 'y3', width: 2,
            marker: {
                visible: true,
                width: 10, height: 10,
                shape: 'Circle'
            }
        }
    ],
    title: 'FB Penetration of Internet Audience',
    legendSettings: {
        width: '180',
        enablePages: false
    }
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>

```



```

</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
</body>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Legend Item Padding

The [itemPadding](#) property can be used to adjust the space between the legend items.

INDEX.JS

```

var chartData = [
    { country: "USA", gold: 50, silver: 70, bronze: 45 },
    { country: "China", gold: 40, silver: 60, bronze: 55 },
    { country: "Japan", gold: 70, silver: 60, bronze: 50 },
    { country: "Australia", gold: 60, silver: 56, bronze: 40 },
    { country: "France", gold: 50, silver: 45, bronze: 35 },
    { country: "Germany", gold: 40, silver: 30, bronze: 22 },
    { country: "Italy", gold: 40, silver: 35, bronze: 37 },
    { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];

var chart = new ej.charts.Chart({
    primaryXAxis: {
        valueType: 'Category',
        title: 'Countries'
    },
    primaryYAxis: {
        minimum: 0, maximum: 80,
        interval: 20, title: 'Medals'
    },
    series:[{
        dataSource: chartData,
        xName: 'country', yName: 'gold',
        name: 'Gold', type: 'Column'
    }, {
        dataSource: chartData,
        xName: 'country', yName: 'silver',
        name: 'Silver', type: 'Column'
    }, {
        dataSource: chartData,
        xName: 'country', yName: 'bronze',
        name: 'Bronze', type: 'Column'
    }],
    title: 'Olympic Medals',
    legendSettings: {
        visible: true,
        //Legend position as top
    }
});

```

```

        itemPadding: 30
    }
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Note: To use legend feature, we need to inject `Legend` using `Chart.Inject(Legend)`.

See Also

- [Customize each shape in legend](#)

Tooltip in ##Platform_Name## Chart control

<!-- markdownlint-disable MD036 -->

Chart will display details about the points through tooltip, when the mouse is moved over the point.

Default tooltip

By default, tooltip is not visible. You can enable the tooltip by setting `enable` property to `true` and by injecting `Tooltip` module using `Chart.Inject(Tooltip)`.

INDEX.JS

```

var chartData = [
  { x: new Date(1975, 0, 1), y: 16, y1: 10, y2: 4.5 },
  { x: new Date(1980, 0, 1), y: 12.5, y1: 7.5, y2: 5 },
  { x: new Date(1985, 0, 1), y: 19, y1: 11, y2: 6.5 },

```

```

    { x: new Date(1990, 0, 1), y: 14.4, y1: 7, y2: 4.4 },
    { x: new Date(1995, 0, 1), y: 11.5, y1: 8, y2: 5 },
    { x: new Date(2000, 0, 1), y: 14, y1: 6, y2: 1.5 },
    { x: new Date(2005, 0, 1), y: 10, y1: 3.5, y2: 2.5 },
    { x: new Date(2010, 0, 1), y: 16, y1: 7, y2: 3.7 }
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        title: 'Years',
        lineStyle: { width: 0 },
        labelFormat: 'Y',
        intervalType: 'Years',
        valueType: 'DateTime',
        edgeLabelPlacement: 'Shift'
    },
    primaryYAxis:
    {
        title: 'Percentage (%)',
        minimum: 0, maximum: 20, interval: 2,
        labelFormat: '{value}%'
    },
    series: [
        {
            type: 'StepLine',
            dataSource: chartData, xName: 'x', yName: 'y',
            width: 2, name: 'China',
            marker: {
                visible: true, width: 10, height: 10
            },
        },
        {
            type: 'StepLine',
            dataSource: chartData, xName: 'x', yName: 'y1',
            width: 2, name: 'Australia',
            marker: {
                visible: true, width: 10, height: 10
            },
        },
        {
            type: 'StepLine',
            dataSource: chartData, xName: 'x', yName: 'y2',
            width: 2, name: 'Japan',
            marker: {
                visible: true, width: 10, height: 10
            },
        },
    ],
    title: 'Unemployment Rates 1975-2010',
    //Tooltip for chart
    tooltip: {enable: true}
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>

```

```

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
    <script id="Unemployment" type="text/x-template">
        <div id='templateWrap'>
            <table style="width:100%; border: 1px solid black;">
                <tr><th colspan="2" bgcolor="#00FFFF">Unemployment</th></tr>
                <tr><td bgcolor="#00FFFF">${x}:</td><td
bgcolor="#00FFFF">${y}</td></tr>
            </table>
        </div>
    </script>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

<!-- markdownlint-disable MD013 -->

Fixed tooltip

By default, tooltip track the mouse movement, but you can set a fixed position for the tooltip by using the [location](#) property.

INDEX.JS

```

var chartData = [
    { x: new Date(1975, 0, 1), y: 16, y1: 10, y2: 4.5 },
    { x: new Date(1980, 0, 1), y: 12.5, y1: 7.5, y2: 5 },
    { x: new Date(1985, 0, 1), y: 19, y1: 11, y2: 6.5 },
    { x: new Date(1990, 0, 1), y: 14.4, y1: 7, y2: 4.4 },
    { x: new Date(1995, 0, 1), y: 11.5, y1: 8, y2: 5 },
    { x: new Date(2000, 0, 1), y: 14, y1: 6, y2: 1.5 },
    { x: new Date(2005, 0, 1), y: 10, y1: 3.5, y2: 2.5 },
    { x: new Date(2010, 0, 1), y: 16, y1: 7, y2: 3.7 }
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        title: 'Years',

```

```

        lineStyle: { width: 0 },
        labelFormat: 'y',
        intervalType: 'Years',
        valueType: 'DateTime',
        edgeLabelPlacement: 'Shift'
    },
    primaryYAxis:
    {
        title: 'Percentage (%)',
        minimum: 0, maximum: 20, interval: 2,
        labelFormat: '{value}%'
    },
    series: [
        {
            type: 'StepLine',
            dataSource: chartData, xName: 'x', yName: 'y',
            width: 2, name: 'China',
            marker: {
                visible: true, width: 10, height: 10
            }
        },
        {
            type: 'StepLine',
            dataSource: chartData, xName: 'x', yName: 'y1',
            width: 2, name: 'Australia',
            marker: {
                visible: true, width: 10, height: 10
            }
        },
        {
            type: 'StepLine',
            dataSource: chartData, xName: 'x', yName: 'y2',
            width: 2, name: 'Japan',
            marker: {
                visible: true, width: 10, height: 10
            }
        }
    ],
    title: 'Unemployment Rates 1975-2010',
    //Tooltip for chart
    tooltip: { enable: true, location: { x: 120, y: 20 } }
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>

```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Format the tooltip

<!-- markdownlint-disable MD013 -->

By default, tooltip shows information of x and y value in points. In addition to that, you can show more information in tooltip. For example the format `${series.name} ${point.x}` shows series name and point x value.

INDEX.JS

```

var chartData = [
    { x: new Date(1975, 0, 1), y: 16, y1: 10, y2: 4.5 },
    { x: new Date(1980, 0, 1), y: 12.5, y1: 7.5, y2: 5 },
    { x: new Date(1985, 0, 1), y: 19, y1: 11, y2: 6.5 },
    { x: new Date(1990, 0, 1), y: 14.4, y1: 7, y2: 4.4 },
    { x: new Date(1995, 0, 1), y: 11.5, y1: 8, y2: 5 },
    { x: new Date(2000, 0, 1), y: 14, y1: 6, y2: 1.5 },
    { x: new Date(2005, 0, 1), y: 10, y1: 3.5, y2: 2.5 },
    { x: new Date(2010, 0, 1), y: 16, y1: 7, y2: 3.7 }
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        title: 'Years',
        lineStyle: { width: 0 },
        labelFormat: 'y',
        intervalType: 'Years',
        valueType: 'DateTime',
        edgeLabelPlacement: 'Shift'
    },
    primaryYAxis: {
        {
            title: 'Percentage (%)',
            minimum: 0, maximum: 20, interval: 2,
            labelFormat: '{value}%'
        },
    },
    series: [
        {
            type: 'StepLine',
            dataSource: chartData, xName: 'x', yName: 'y',
            width: 2, name: 'China',

```

```

        marker: {
            visible: true, width: 10, height: 10
        },
    },
    {
        type: 'StepLine',
        dataSource: chartData, xName: 'x', yName: 'y1',
        width: 2, name: 'Australia',
        marker: {
            visible: true, width: 10, height: 10
        },
    },
    {
        type: 'StepLine',
        dataSource: chartData, xName: 'x', yName: 'y2',
        width: 2, name: 'Japan',
        marker: {
            visible: true, width: 10, height: 10
        },
    },
],
title: 'Unemployment Rates 1975-2010',
tooltip: {
    enable: true,
    //tooltip format for chart
    format: '${series.name} ${point.x} : ${point.y}'
}
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
    <script id="Unemployment" type="text/x-template">
        <div id='templateWrap'>
            <table style="width:100%; border: 1px solid black;">
                <tr><th colspan="2" bgcolor="#00FFFF">Unemployment</th></tr>
                <tr><td bgcolor="#00FFFF">${x}</td><td
bgcolor="#00FFFF">${y}</td></tr>

```

```

    </table>
  </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Tooltip template

Any HTML elements can be displayed in the tooltip by using the [template](#) property of the tooltip. You can use the `${x}` and `${y}` as place holders in the HTML element to display the x and y values of the corresponding data point.

INDEX.JS

```

var chartData = [
  { x: 1975, y: 16, y1: 10, y2: 4.5 },
  { x: 1980, y: 12.5, y1: 7.5, y2: 5 },
  { x: 1985, y: 19, y1: 11, y2: 6.5 },
  { x: 1990, y: 14.4, y1: 7, y2: 4.4 },
  { x: 1995, y: 11.5, y1: 8, y2: 5 },
  { x: 2000, y: 14, y1: 6, y2: 1.5 },
  { x: 2005, y: 10, y1: 3.5, y2: 2.5 },
  { x: 2010, y: 16, y1: 7, y2: 3.7 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    title: 'Years',
    lineStyle: { width: 0 },
    minimum: 1975,
    maximum: 2010,
    interval: 5,
    edgeLabelPlacement: 'Shift'
  },
  primaryYAxis: {
    {
      title: 'Percentage (%)',
      minimum: 0, maximum: 20, interval: 2,
      labelFormat: '{value}%'
    },
    series: [
      {
        type: 'StepLine',
        dataSource: chartData, xName: 'x', yName: 'y',
        width: 2, name: 'China',
        marker: {
          visible: true, width: 10, height: 10
        },
      },
      {
        type: 'StepLine',
        dataSource: chartData, xName: 'x', yName: 'y1',

```



```

        width: 2, name: 'Australia',
        marker: {
            visible: true, width: 10, height: 10
        },
    },
    {
        type: 'StepLine',
        dataSource: chartData, xName: 'x', yName: 'y2',
        width: 2, name: 'Japan',
        marker: {
            visible: true, width: 10, height: 10
        },
    },
],
title: 'Unemployment Rates 1975-2010',
tooltip: {
    enable: true,
    //tooltip template for chart
    template: '<div>${x}</div><div>${y}</div>'
}
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
    <script id="Unemployment" type="text/x-template">
        <div id='templateWrap'>
            <table style="width:100%; border: 1px solid black;">
                <tr><th colspan="2" bgcolor="#00FFFF">Unemployment</th></tr>
                <tr><td bgcolor="#00FFFF">${x}</td><td
bgcolor="#00FFFF">${y}</td></tr>
            </table>
        </div>
    </script>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";

```

```

}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customize the appearance of tooltip

The [fill](#) and [border](#) properties are used to customize the background color and border of the tooltip respectively. The [textStyle](#) property in the tooltip is used to customize the font of the tooltip text. The [highlightColor](#) property is used to customize the point color while hovering for tooltip.

INDEX.JS

```

var chartData = [
  { x: new Date(1975, 0, 1), y: 16, y1: 10, y2: 4.5 },
  { x: new Date(1980, 0, 1), y: 12.5, y1: 7.5, y2: 5 },
  { x: new Date(1985, 0, 1), y: 19, y1: 11, y2: 6.5 },
  { x: new Date(1990, 0, 1), y: 14.4, y1: 7, y2: 4.4 },
  { x: new Date(1995, 0, 1), y: 11.5, y1: 8, y2: 5 },
  { x: new Date(2000, 0, 1), y: 14, y1: 6, y2: 1.5 },
  { x: new Date(2005, 0, 1), y: 10, y1: 3.5, y2: 2.5 },
  { x: new Date(2010, 0, 1), y: 16, y1: 7, y2: 3.7 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    title: 'Years',
    lineStyle: { width: 0 },
    labelFormat: 'y',
    intervalType: 'Years',
    valueType: 'DateTime',
    edgeLabelPlacement: 'Shift'
  },
  //Highlight color for tooltip
  highlightColor: 'red',
  primaryYAxis: {
    {
      title: 'Percentage (%)',
      minimum: 0, maximum: 20, interval: 2,
      labelFormat: '{value}%'
    }
  },
  series: [
    {
      type: 'StepLine',
      dataSource: chartData, xName: 'x', yName: 'y',
      width: 2, name: 'China',
      marker: {
        visible: true, width: 10, height: 10
      }
    },
    {
      type: 'StepLine',
      dataSource: chartData, xName: 'x', yName: 'y1',
      width: 2, name: 'Australia',
      marker: {
        visible: true, width: 10, height: 10
      }
    }
  ]
});

```

```

        {
            type: 'StepLine',
            dataSource: chartData, xName: 'x', yName: 'y2',
            width: 2, name: 'Japan',
            marker: {
                visible: true, width: 10, height: 10
            },
        },
    ],
    title: 'Unemployment Rates 1975-2010',
    tooltip: {
        enable: true,
        format: '${series.name} ${point.x} : ${point.y}',
        //fill for tooltip
        fill: '#7bb4eb',
        //border for tooltip
        border: {
            width: 2,
            color: 'grey'
        }
    }
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
    <script id="Unemployment" type="text/x-template">
        <div id='templateWrap'>
            <table style="width:100%; border: 1px solid black;">
                <tr><th colspan="2" bgcolor="#00FFFF">Unemployment</th></tr>
                <tr><td bgcolor="#00FFFF">${x}</td><td
bgcolor="#00FFFF">${y}</td></tr>
            </table>
        </div>
    </script>
<script>
var ele = document.getElementById('container');
if(ele) {

```

```

    ele.style.visibility = "visible";
  }
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See also

- [Format the tooltip value](#)
- [Create a table in tooltip](#)

Zooming in ##Platform_Name## Chart control

Enable zooming

Chart can be zoomed in three ways.

- Selection - By setting [enableSelectionZooming](#) property to true in `zoomSettings`, you can zoom the chart by using the rubber band selection.
- Mousewheel - By setting [enableMouseWheelZooming](#) property to true in `zoomSettings`, you can zoom in and zoom out the chart by scrolling the mouse wheel.
- Pinch - By setting [enablePinchZooming](#) property to true in `zoomSettings`, you can zoom the chart through pinch gesture in touch enabled devices.

Pinch zooming is supported only in browsers that support multi-touch gestures. Currently IE11, Chrome and Opera browsers support multi-touch in desktop devices.

INDEX.JS

```

var series1 = [];
var point1;
var value = 80;
var i;
for (i = 1; i < 500; i++) {
  if (Math.random() > .5) {
    value += Math.random();
  } else {
    value -= Math.random();
  }
  point1 = { x: new Date(1950, i + 2, i), y: value.toFixed(1) };
  series1.push(point1);
}
var chart = new ej.charts.Chart({
  chartArea : {border : {width : 0}},
  primaryXAxis: {
    title: 'Years',
    valueType: 'DateTime',
    labelFormat: 'yMMM',
    edgeLabelPlacement: 'Shift',
    majorGridLines : { width : 0 }
  },
  primaryYAxis:
  {
    title: 'Profit ($)',

```

```

        rangePadding: 'None',
        lineStyle : { width: 0 },
        majorTickLines : {width : 0}
    },
    series: [
        {
            type: 'Area',
            dataSource: series1,
            name: 'Product X',
            xName: 'x',
            yName: 'y',
            border: { width: 0.5, color: '#00bdae' },
            animation: { enable: false }
        },
    ],
    //Zooming for chart
    zoomSettings:
    {
        enableMouseWheelZooming: true,
        enablePinchZooming: true,
        enableSelectionZooming: true
    },
    title: 'Sales History of Product X',
    legendSettings: { visible: false },
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
    <script id="Unemployment" type="text/x-template">
        <div id='templateWrap'>
            <table style="width:100%; border: 1px solid black;">
                <tr><th colspan="2" bgcolor="#00FFFF">Unemployment</th></tr>
                <tr><td bgcolor="#00FFFF">${x}</td><td
            bgcolor="#00FFFF">${y}</td></tr>
            </table>
        </div>
    </script>

```

```
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

After zooming the chart, a zooming toolbar will appear with **zoom**, **zoomin**, **zoomout**, **pan** and **reset** buttons. Selecting the Pan option will allow to pan the chart and selecting the Reset option will reset the zoomed chart.

Modes

The [mode](#) property in zoomSettings specifies whether the chart is allowed to scale along the horizontal axis or vertical axis. The default value of the mode is XY (both axis).

There are three types of mode.

- X- Allows us to zoom the chart horizontally.
- Y - Allows us to zoom the chart vertically.
- XY – Allows us to zoom the chart both vertically and horizontally.

INDEX.JS

```
var series1 = [];
var point1;
var value = 80;
var i;
for (i = 1; i < 500; i++) {
    if (Math.random() > .5) {
        value += Math.random();
    } else {
        value -= Math.random();
    }
    point1 = { x: new Date(1950, i + 2, i), y: value.toFixed(1) };
    series1.push(point1);
}
var chart = new ej.charts.Chart({
    chartArea : {border : {width : 0}},
    primaryXAxis: {
        title: 'Years',
        valueType: 'DateTime',
        labelFormat: 'yMMM',
        edgeLabelPlacement: 'Shift',
        majorGridLines : { width : 0 }
    },
    primaryYAxis:
    {
        title: 'Profit ($)',
        rangePadding: 'None',
        lineStyle : { width: 0 },
        majorTickLines : {width : 0}
    },
    series: [
        {
            name: 'Profit',
            type: 'line',
            dataSource: series1,
            markers: {
                visible: true,
                size: 10,
                fill: 'white',
                stroke: 'black',
                strokeWidth: 1
            }
        }
    ]
});
```

```

series: [
    {
        type: 'Area',
        dataSource: series1,
        name: 'Product X',
        xName: 'x',
        yName: 'y',
        border: { width: 0.5, color: '#00bdae' },
        animation: { enable: false }
    },
],
zoomSettings:
{
    enableSelectionZooming: true,
    //zoom mode as x
    mode: 'X'
},
title: 'Sales History of Product X',
legendSettings: { visible: false },
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script id="Unemployment" type="text/x-template">
    <div id='templateWrap'>
      <table style="width:100%; border: 1px solid black;">
        <tr><th colspan="2" bgcolor="#00FFFF">Unemployment</th></tr>
        <tr><td bgcolor="#00FFFF">${x}</td><td
bgcolor="#00FFFF">${y}</td></tr>
      </table>
    </div>
  </script>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}

```

```

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Toolbar

By default, zoomin, zoomout, pan and reset buttons will be displayed for zoomed chart. You can customize to show the desired options in the toolbar using the [toolbarItems](#) property. Also using the [showToolbar](#) property, you can show toolkit for zooming and panning the chart during initial rendering itself.

INDEX.JS

```

var series1 = [];
var point1;
var value = 80;
var i;
for (i = 1; i < 500; i++) {
    if (Math.random() > .5) {
        value += Math.random();
    } else {
        value -= Math.random();
    }
    point1 = { x: new Date(1950, i + 2, i), y: value.toFixed(1) };
    series1.push(point1);
}
var chart = new ej.charts.Chart({
    chartArea : {border : {width : 0}},
    primaryXAxis: {
        title: 'Years',
        valueType: 'DateTime',
        labelFormat: 'yMMM',
        edgeLabelPlacement: 'Shift',
        majorGridLines : { width : 0 }
    },
    primaryYAxis:
    {
        title: 'Profit ($)',
        rangePadding: 'None',
        lineStyle : { width: 0 },
        majorTickLines : {width : 0}
    },
    series: [
        {
            type: 'Area',
            dataSource: series1,
            name: 'Product X',
            xName: 'x',
            yName: 'y',
            border: { width: 0.5, color: '#00bdae' },
            animation: { enable: false }
        },
    ],
    zoomSettings:
    {
        enableSelectionZooming: true,
        //toolbar items for zooming toolkit
    }
});

```



```

        toolbarItems: ['Zoom', 'Pan', 'Reset'],
        showToolbar: true
    },
    title: 'Sales History of Product X',
    legendSettings: { visible: false },
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script id="Unemployment" type="text/x-template">
    <div id='templateWrap'>
      <table style="width:100%; border: 1px solid black;">
        <tr><th colspan="2" bgcolor="#00FFFF">Unemployment</th></tr>
        <tr><td bgcolor="#00FFFF">${x}</td><td
        bgcolor="#00FFFF">${y}</td></tr>
      </table>
    </div>
  </script>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Enable pan

Using [enablePan](#) property you can able to pan the zoomed chart without help of toolbar items.

INDEX.JS

```

var series1 = [];
var point1;
var value = 80;
var i;
for (i = 1; i < 500; i++) {

```

```

    if (Math.random() > .5) {
        value += Math.random();
    } else {
        value -= Math.random();
    }
    point1 = { x: new Date(1950, i + 2, i), y: value.toFixed(1) };
    series1.push(point1);
}
var chart = new ej.charts.Chart({
    chartArea : {border : {width : 0}},
    primaryXAxis: {
        title: 'Years',
        valueType: 'DateTime',
        labelFormat: 'yMMM',
        edgeLabelPlacement: 'Shift',
        majorGridLines : { width : 0 }
    },
    primaryYAxis:
    {
        title: 'Profit ($)',
        rangePadding: 'None',
        lineStyle : { width: 0 },
        majorTickLines : {width : 0}
    },
    series: [
        {
            type: 'Area',
            dataSource: series1,
            name: 'Product X',
            xName: 'x',
            yName: 'y',
            border: { width: 0.5, color: '#00bdae' },
            animation: { enable: false }
        },
    ],
    zoomSettings:
    {
        enableSelectionZooming: true,
        //toolbar items for zooming toolkit
        toolbarItems: ['Zoom', 'Pan', 'Reset'],
        showToolbar: true
    },
    title: 'Sales History of Product X',
    legendSettings: { visible: false },
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
    <script id="Unemployment" type="text/x-template">
        <div id='templateWrap'>
            <table style="width:100%; border: 1px solid black;">
                <tr><th colspan="2" bgcolor="#00FFFF">Unemployment</th></tr>
                <tr><td bgcolor="#00FFFF">${x}</td><td
bgcolor="#00FFFF">${y}</td></tr>
            </table>
        </div>
    </script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Enable scrollbar

Using the [enableScrollbar](#) property, you can add a scrollbar to a zoomed chart. This scrollbar allows you to zoom or pan the chart. The appearance of the scrollbar can be customized using properties in [scrollbarSettings](#). For example, you can use [trackColor](#) and [trackRadius](#) properties to customize the track of the scrollbar, and [scrollbarRadius](#) and [scrollbarColor](#) properties to customize the scroller. The ability to zoom through the scrollbar can be enabled or disabled using the [enableZoom](#) property in [scrollbarSettings](#). Additionally, you can change the color of the grip and height of the scrollbar using the [gripColor](#) and [height](#) properties.

INDEX.JS

```

var series1 = [];
var point1;
var value = 80;
var i;
for (i = 1; i < 500; i++) {
    if (Math.random() > .5) {
        value += Math.random();
    } else {
        value -= Math.random();
    }
    point1 = { x: new Date(1950, i + 2, i), y: value.toFixed(1) };
    series1.push(point1);
}
var chart = new ej.charts.Chart({
    chartArea: { border: { width: 0 } },

```

```

primaryXAxis: {
  valueType: 'DateTime',
  zoomFactor: 0.2,
  zoomPosition: 0.6,
  scrollbarSettings: {
    enable: true,
    enableZoom: false,
    height: 14,
    trackRadius: 8,
    scrollbarRadius: 8,
    gripColor: 'transparent',
    trackColor: 'yellow',
    scrollbarColor: 'red'
  }
},
series: [
  {
    type: 'Area',
    dataSource: series1,
    name: 'Product X',
    xName: 'x',
    yName: 'y',
    border: { width: 0.5, color: '#00bdae' },
    animation: { enable: false }
  },
],
//Zooming for chart
zoomSettings: {
  enableSelectionZooming: true,
  enableScrollbar: true,
  mode: 'X'
},
title: 'Sales History of Product X',
legendSettings: { visible: false },
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">

```

```

        <div id="element"></div>
    </div>
    <script id="Unemployment" type="text/x-template">
        <div id='templateWrap'>
            <table style="width:100%; border: 1px solid black;">
                <tr><th colspan="2" bgcolor="#00FFFF">Unemployment</th></tr>
                <tr><td bgcolor="#00FFFF">${x}</td><td
bgcolor="#00FFFF">${y}</td></tr>
            </table>
        </div>
    </script>
    <script>
    var ele = document.getElementById('container');
    if(ele) {
        ele.style.visibility = "visible";
    }
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

Auto interval on zooming

By using [enableAutoIntervalOnZooming](#) property, the axis interval will get calculated automatically with respect to the zoomed range.

INDEX.JS

```

var series1 = [];
var point1;
var value = 80;
var i;
for (i = 1; i < 500; i++) {
    if (Math.random() > .5) {
        value += Math.random();
    } else {
        value -= Math.random();
    }
    point1 = { x: new Date(1950, i + 2, i), y: value.toFixed(1) };
    series1.push(point1);
}
var chart = new ej.charts.Chart({
    chartArea: {border: {width: 0}},
    primaryXAxis: {
        title: 'Years',
        valueType: 'DateTime',
        labelFormat: 'yMMM',
        edgeLabelPlacement: 'Shift',
        majorGridLines: { width: 0 },
        enableAutoIntervalOnZooming: true
    },
    primaryYAxis:
    {
        title: 'Profit ($)',
        rangePadding: 'None',
        lineStyle: { width: 0 },
        majorTickLines: {width: 0},
        enableAutoIntervalOnZooming: true
    }
});

```

```

    },
    series: [
        {
            type: 'Area',
            dataSource: series1,
            name: 'Product X',
            xName: 'x',
            yName: 'y',
            border: { width: 0.5, color: '#00bdae' },
            animation: { enable: false }
        },
    ],
    zoomSettings:
    {
        enableSelectionZooming: true,
        //toolbar items for zooming toolkit
        toolbarItems: ['Zoom', 'Pan', 'Reset']
    },
    title: 'Sales History of Product X',
    legendSettings: { visible: false },
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
    <script id="Unemployment" type="text/x-template">
        <div id='templateWrap'>
            <table style="width:100%; border: 1px solid black;">
                <tr><th colspan="2" bgcolor="#00FFFF">Unemployment</th></tr>
                <tr><td bgcolor="#00FFFF">${x}:</td><td
bgcolor="#00FFFF">${y}</td></tr>
            </table>
        </div>
    </script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";

```

```

}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Note: To use zooming feature, we need to inject `Zoom` module `Chart.Inject(Zoom)` method.

<!-- markdownlint-disable MD036 -->

Data editing in ##Platform_Name## Chart control

Enable Data Editing

We can use the data editing through inject the `DataEditing` module in the chart. It provides drag and drop support to the rendered points. Now, we can change the location or value of the point based on its `y` value. To enable the data editing, set the `enable` property to true in the drag settings of the series. Also, we can set color using `fill` property and set the data editing minimum and maximum range using `minY` and `maxY` properties.

INDEX.JS

```

var chart = new ej.charts.Chart({
  primaryXAxis: {
    valueType: 'DateTime',
    labelFormat: 'y',
    intervalType: 'Years',
    edgeLabelPlacement: 'Shift',
    majorGridLines: { width: 0 }
  },
  //Initializing Primary Y Axis
  primaryYAxis: {
    {
      labelFormat: '{value}%',
      rangePadding: 'None',
      minimum: 0,
      maximum: 100,
      interval: 20,
      lineStyle: { width: 0 },
      majorTickLines: { width: 0 },
      minorTickLines: { width: 0 }
    }
  },
  chartArea: {
    border: {
      width: 0
    }
  },
  //Initializing Chart Series
  series: [
    {
      type: 'Column',
      dataSource: [
        { x: new Date(2005, 0, 1), y: 21 }, { x: new Date(2006, 0,
1), y: 24 },
        { x: new Date(2007, 0, 1), y: 36 }, { x: new Date(2008, 0,
1), y: 38 },
        { x: new Date(2009, 0, 1), y: 54 }, { x: new Date(2010, 0,
1), y: 57 },

```

```

        { x: new Date(2011, 0, 1), y: 70 }
    ],
    xName: 'x', width: 2, marker: {
        visible: true,
        width: 10,
        height: 10
    },
    yName: 'y', name: 'Germany', dragSettings: { enable: true, }
},
{
    type: 'Line',
    dataSource: [
        { x: new Date(2005, 0, 1), y: 21 }, { x: new Date(2006, 0,
1), y: 24 },
        { x: new Date(2007, 0, 1), y: 36 }, { x: new Date(2008, 0,
1), y: 38 },
        { x: new Date(2009, 0, 1), y: 54 }, { x: new Date(2010, 0,
1), y: 57 },
        { x: new Date(2011, 0, 1), y: 70 }
    ],
    xName: 'x', width: 2, marker: {
        visible: true,
        width: 10,
        height: 10
    },
    yName: 'y', name: 'Germany', dragSettings: { enable: true, }
}
],
//Initializing Chart title
title: 'Inflation - Consumer Price',
//Initializing User Interaction Tooltip
tooltip: {
    enable: true
},
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>

```



```

</div>
<script id="Unemployment" type="text/x-template">
  <div id='templateWrap'>
    <table style="width:100%; border: 1px solid black;">
      <tr><th colspan="2" bgcolor="#00FFFF">Unemployment</th></tr>
      <tr><td bgcolor="#00FFFF">${x}</td><td
bgcolor="#00FFFF">${y}</td></tr>
    </table>
  </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Cross hair and track ball in ##Platform_Name## Chart control

Crosshair has a vertical and horizontal line to view the value of the axis at mouse or touch position.

Crosshair lines can be enabled by using [enable](#) property in the `crosshair`. Likewise tooltip label for an axis can be enabled by using [enable](#) property of `crosshairTooltip` in the corresponding axis.

INDEX.JS

```

var series1 = [];
var series2 = [];
var point1;
var point2;
var value = 60;
var value1 = 50;
var i;
for (i = 1; i < 250; i++) {
  if (Math.random() > .5) {
    value += Math.random();
    value1 += Math.random();
  } else {
    value -= Math.random();
    value1 -= Math.random();
  }
  point1 = { x: new Date(2000, i, 1), y: value };
  point2 = { x: new Date(2000, i, 1), y: value1 };
  series1.push(point1);
  series2.push(point2);
}
var char = new ej.charts.Chart({
  primaryXAxis: {
    majorGridLines: { width: 0 },
    valueType: 'DateTime',
    crosshairTooltip: { enable: true },
    labelFormat: 'yMMM'
  },
  primaryYAxis:
{

```

```

        minimum: 10, maximum: 90, interval: 10,
        title: 'Temperature (°F)',
        rowIndex: 0,
        crosshairTooltip: { enable: true }
    },
    axes: [
        {
            majorGridLines: { width: 0 },
            rowIndex: 0, opposedPosition: true,
            minimum: 0, maximum: 160, interval: 20,
            name: 'yAxis', title: 'Rainfall (MM)',
            crosshairTooltip: { enable: true }
        }
    ],
    series: [
        {
            type: 'Line', width: 2, name: 'Temperature',
            dataSource: series1, xName: 'x', yName: 'y'
        },
        {
            type: 'Line', name: 'Rainfall', width: 2,
            dataSource: series2, xName: 'x', yName: 'y',
            yAxisName: 'yAxis'
        }
    ],
    //crosshair for chart
    crosshair: { enable: true },
    legendSettings: { visible: true },
    title: 'Weather Condition'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
    <script id="Unemployment" type="text/x-template">
        <div id='templateWrap'>
            <table style="width:100%; border: 1px solid black;">
            <tr><th colspan="2" bgcolor="#00FFFF">Unemployment</th></tr>

```

```

        <tr><td bgcolor="#00FFFF">${x}</td><td
        bgcolor="#00FFFF">${y}</td></tr>
    </table>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Tooltip for axis

Tooltip label for an axis can be enabled by using [enable](#) property of `crosshairTooltip` in the corresponding axis.

INDEX.JS

```

var series1 = [];
var series2 = [];
var point1;
var point2;
var value = 60;
var value1 = 50;
var i;
for (i = 1; i < 250; i++) {
    if (Math.random() > .5) {
        value += Math.random();
        value1 += Math.random();
    } else {
        value -= Math.random();
        value1 -= Math.random();
    }
    point1 = { x: new Date(2000, i, 1), y: value };
    point2 = { x: new Date(2000, i, 1), y: value1 };
    series1.push(point1);
    series2.push(point2);
}
var char = new ej.charts.Chart({
    primaryXAxis: {
        majorGridLines: { width: 0 },
        valueType: 'DateTime',
        crosshairTooltip: { enable: true },
        labelFormat: 'yMMM'
    },
    primaryYAxis:
    {
        minimum: 10, maximum: 90, interval: 10,
        title: 'Temperature (°F)',
        rowIndex: 0,
        crosshairTooltip: { enable: true }
    },
    axes: [
        {

```

```

        majorGridLines: { width: 0 },
        rowIndex: 0, opposedPosition: true,
        minimum: 0, maximum: 160, interval: 20,
        name: 'yAxis', title: 'Rainfall (MM)',
        crosshairTooltip: { enable: true }
    },
    ],
    series: [
        {
            type: 'Line', width: 2, name: 'Temperature',
            dataSource: series1, xName: 'x', yName: 'y'
        },
        {
            type: 'Line', name: 'Rainfall', width: 2,
            dataSource: series2, xName: 'x', yName: 'y',
            yAxisName: 'yAxis'
        }
    ],
    //crosshair for chart
    crosshair: { enable: true },
    legendSettings: { visible: true },
    title: 'Weather Condition'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
    <script id="Unemployment" type="text/x-template">
        <div id='templateWrap'>
            <table style="width:100%; border: 1px solid black;">
            <tr><th colspan="2" bgcolor="#00FFFF">Unemployment</th></tr>
            <tr><td bgcolor="#00FFFF">${x}</td><td
            bgcolor="#00FFFF">${y}</td></tr>
            </table>
        </div>
    </script>
    <script>
    var ele = document.getElementById('container');

```

```

if(ele) {
    ele.style.visibility = "visible";
}

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customization

The [fill](#) and [textStyle](#) property of the `crosshairTooltip` is used to customize the background color and font style of the crosshair label respectively. Color and width of the crosshair line can be customized by using the [line](#) property in the crosshair.

INDEX.JS

```

var series1 = [];
var series2 = [];
var point1;
var point2;
var value = 60;
var value1 = 50;
var i;
for (i = 1; i < 250; i++) {
    if (Math.random() > .5) {
        value += Math.random();
        value1 += Math.random();
    } else {
        value -= Math.random();
        value1 -= Math.random();
    }
    point1 = { x: new Date(2000, i, 1), y: value };
    point2 = { x: new Date(2000, i, 1), y: value1 };
    series1.push(point1);
    series2.push(point2);
}
var chart = new ej.charts.Chart({
    primaryXAxis: {
        majorGridLines: { width: 0 },
        valueType: 'DateTime',
        crosshairTooltip: { enable: true, fill: 'green' },
        labelFormat: 'yMMM'
    },
    primaryYAxis: {
        {
            minimum: 10, maximum: 90, interval: 10,
            title: 'Temperature (°F)',
            rowIndex: 0,
            crosshairTooltip: { enable: true, fill: 'green' },
        },
    },
    axes: [
        {
            majorGridLines: { width: 0 },
            rowIndex: 0, opposedPosition: true,
            minimum: 0, maximum: 160, interval: 20,
            name: 'yAxis', title: 'Rainfall (MM)',
            crosshairTooltip: { enable: true, fill: 'green' },
        }
    ]
});

```

```

    ],
    series: [
        {
            type: 'Line', width: 2, name: 'Temperature',
            dataSource: series1, xName: 'x', yName: 'y'
        },
        {
            type: 'Line', name: 'Rainfall', width: 2,
            dataSource: series2, xName: 'x', yName: 'y',
            yAxisName: 'yAxis'
        }
    ],
    crosshair: {
        enable: true,
        //customizing crosshair
        line: {width: 2, color: 'green'} },
    legendSettings: { visible: true },
    title: 'Weather Condition'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
    <script id="Unemployment" type="text/x-template">
        <div id='templateWrap'>
            <table style="width:100%; border: 1px solid black;">
                <tr><th colspan="2" bgcolor="#00FFFF">Unemployment</th></tr>
                <tr><td bgcolor="#00FFFF">${x}</td><td
bgcolor="#00FFFF">${y}</td></tr>
            </table>
        </div>
    </script>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>

```

```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Note: To use crosshair feature, we need to inject **Crosshair** module **Chart.Inject(Crosshair)** method.

Trackball

Trackball is used to track a data point closest to the mouse or touch position. Trackball marker indicates the closest point and trackball tooltip displays the information about the point. To use trackball feature, we need to inject **Crosshair** module and **Tooltip** module using

Chart.Inject(Crosshair, Tooltip).

Trackball can be enabled by setting the [enable](#) property of the crosshair to true and [shared](#) property in **tooltip** to true in chart.

INDEX.JS

```
var chartData = [
  { x: new Date(2000, 2, 11), y: 15, y1: 39, y2: 60, y3: 75, y4: 85 },
  { x: new Date(2000, 9, 14), y: 20, y1: 30, y2: 55, y3: 75, y4: 83 },
  { x: new Date(2001, 2, 11), y: 25, y1: 28, y2: 48, y3: 68, y4: 85 },
  { x: new Date(2001, 9, 16), y: 21, y1: 35, y2: 57, y3: 75, y4: 87 },
  { x: new Date(2002, 2, 7), y: 13, y1: 39, y2: 62, y3: 71, y4: 82 },
  { x: new Date(2002, 9, 7), y: 18, y1: 41, y2: 64, y3: 69, y4: 74 },
  { x: new Date(2003, 2, 11), y: 24, y1: 45, y2: 57, y3: 81, y4: 73 },
  { x: new Date(2003, 9, 14), y: 23, y1: 48, y2: 53, y3: 84, y4: 75 },
  { x: new Date(2004, 2, 6), y: 19, y1: 54, y2: 63, y3: 85, y4: 73 },
  { x: new Date(2004, 9, 6), y: 31, y1: 55, y2: 50, y3: 87, y4: 60 },
  { x: new Date(2005, 2, 11), y: 39, y1: 57, y2: 66, y3: 75, y4: 48 },
  { x: new Date(2005, 9, 11), y: 50, y1: 60, y2: 65, y3: 70, y4: 55 },
  { x: new Date(2006, 2, 11), y: 24, y1: 60, y2: 79, y3: 85, y4: 40 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    title: 'Years',
    minimum: new Date(2000, 1, 1), maximum: new Date(2006, 2, 11),
    intervalType: 'Years',
    valueType: 'DateTime',
    lineStyle: { width: 0 },
    majorGridLines: { width: 0 },
    edgeLabelPlacement: 'Shift'
  },
  primaryYAxis: {
    {
      title: 'Revenue in Millions',
      labelFormat: '{value}M',
      majorTickLines: { width: 0 },
      minimum: 10, maximum: 90,
      lineStyle: { width: 0 }
    },
  },
  series: [
    {
      dataSource: chartData, name: 'John', xName: 'x',
      marker: { visible: true },
      type: 'Line', width: 2,

```

```

        yName: 'y'
    },
    {
        dataSource: chartData, name: 'Andrew', xName: 'x',
        marker: { visible: true },
        type: 'Line', width: 2,
        yName: 'y1'
    },
    {
        dataSource: chartData, name: 'Thomas', xName: 'x',
        marker: { visible: true },
        type: 'Line', width: 2,
        yName: 'y2'
    },
    {
        dataSource: chartData, name: 'Mark', xName: 'x',
        marker: { visible: true },
        type: 'Line', width: 2,
        yName: 'y3'
    },
    {
        dataSource: chartData, name: 'William', xName: 'x',
        marker: { visible: true },
        type: 'Line', width: 2,
        yName: 'y4'
    }
],
// trackball for chart
tooltip: { enable: true, shared: true, format: '${series.name} :
${point.x} : ${point.y}' },
crosshair: { enable: true, lineType: 'Vertical' },
title: 'Average Sales per Person'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script id="Unemployment" type="text/x-template">

```



```

<div id='templateWrap'>
  <table style="width:100%; border: 1px solid black;">
    <tr><th colspan="2" bgcolor="#00FFFF">Unemployment</th></tr>
    <tr><td bgcolor="#00FFFF">${x}</td><td
bgcolor="#00FFFF">${y}</td></tr>
  </table>
</div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Synchronized Charts in ##Platform_Name## Chart control

Tooltip synchronization

The tooltip can be synchronized across multiple charts using the [showTooltip](#) and [hideTooltip](#) methods. When we hover over a data point in one chart, we call the [showTooltip](#) method for the other charts to display related information in other connected charts simultaneously.

In the [showTooltip](#) method, specify the following parameters programmatically to enable tooltip for a particular chart:

- **x** - Data point x-value or x-coordinate value.
- **y** - Data point y-value or y-coordinate value.

INDEX.JS

```

var charts = [];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    minimum: new Date(2023, 1, 18),
    maximum: new Date(2023, 7, 18),
    valueType: 'DateTime',
    labelFormat: 'MMM d',
    lineStyle: { width: 0 },
    majorGridLines: { width: 0 },
    edgeLabelPlacement: ej.base.Browser.isDevice ? 'None' : 'Shift',
    labelRotation: ej.base.Browser.isDevice ? -45 : 0,
    interval: ej.base.Browser.isDevice ? 2 : 1
  },
  primaryYAxis: {
    labelFormat: 'n2',
    majorTickLines: { width: 0 },
    lineStyle: { width: 0 },
    minimum: 0.86,
    maximum: 0.96,
    interval: 0.025
  },
  chartArea: { border: { width: 0 } },

```

```

series: [
    {
        type: 'Line', dataSource: synchronizedData, xName: 'USD', width:
2, yName: 'EUR', emptyPointSettings: { mode: 'Drop' }
    }
],
chartMouseLeave: function (args) {
    chart1.hideTooltip();
},
chartMouseMove: function (args) {
    if ((!ej.base.Browser.isDevice && !chart.isTouch &&
!chart.isChartDrag) || chart.startMove) {
        chart1.startMove = chart.startMove;
        chart1.showTooltip(args.x, args.y);
    }
},
chartMouseUp: function (args) {
    if (ej.base.Browser.isDevice && chart.startMove) {
        chart1.hideTooltip();
    }
},
title: 'US to EURO',
titleStyle: { textAlignment: 'Near' },
tooltip: { enable: true, fadeOutDuration: ej.base.Browser.isDevice ?
2500 : 1000, shared: true, header: '', format: '<b>€${point.y}</b> <br>
${point.x} 2023', enableMarker: false },
});
chart.appendTo('#container1');
charts.push(chart);
var chart1 = new ej.charts.Chart({
    primaryXAxis: {
        minimum: new Date(2023, 1, 18),
        maximum: new Date(2023, 7, 18),
        valueType: 'DateTime',
        labelFormat: 'MMM d',
        lineStyle: { width: 0 },
        majorGridLines: { width: 0 },
        edgeLabelPlacement: ej.base.Browser.isDevice ? 'None' : 'Shift',
        labelRotation: ej.base.Browser.isDevice ? -45 : 0,
        interval: ej.base.Browser.isDevice ? 2 : 1
    },
    primaryYAxis: {
        labelFormat: 'n1',
        majorTickLines: { width: 0 },
        lineStyle: { width: 0 },
        minimum: 79,
        maximum: 85,
        interval: 1.5
    },
    chartArea: { border: { width: 0 } },
    series: [
        {
            type: 'Area', dataSource: synchronizedData, xName: 'USD', width:
2, yName: 'INR', opacity: 0.6, border: { width: 2 }
        }
    ],
    chartMouseMove: function (args) {

```

```

        if ((!ej.base.Browser.isDevice && !chart1.isTouch &&
!chart1.isChartDrag) || chart1.startMove) {
            chart.startMove = chart1.startMove;
            chart.showTooltip(args.x, args.y);
        }
    },
    chartMouseLeave: function (args) {
        chart.hideTooltip();
    },
    chartMouseUp: function (args) {
        if (ej.base.Browser.isDevice && chart1.startMove) {
            chart.hideTooltip();
        }
    },
    title: 'US to INR',
    titleStyle: { textAlign: 'Near' },
    tooltip: { enable: true, fadeOutDuration: ej.base.Browser.isDevice ?
2500 : 1000, shared: true, header: '', format: '<b>₹${point.y}</b> <br>
${point.x} 2023', enableMarker: false },
});
chart1.appendTo('#container2');
charts.push(chart1);

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <script src="es5-datasource.js" type="text/javascript"></script>

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<style>
    #control-container {
        padding: 1px !important;
    }
    .row {
        display: flex;
    }
    .col {
        width: 50%;
        margin: 10px;
        height: 270px;
    }
</style>
<body>

    <div class="control-section">
        <div class="row">

```

```

        <div class="col" id="container1"></div>
        <div class="col" id="container2"></div>
    </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Crosshair synchronization

The crosshair can be synchronized across multiple charts using the [showCrosshair](#) and [hideCrosshair](#) methods. When we hover over one chart, we call the [showCrosshair](#) method for the other charts to align with data points in other connected charts, simplifying data comparison and analysis.

In the [showCrosshair](#) method, specify the following parameters programmatically to enable crosshair for a particular chart:

- **x** - Specifies the x-value of the point or x-coordinate.
- **y** - Specifies the y-value of the point or y-coordinate.

INDEX.JS

```

var charts = [];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        minimum: new Date(2023, 1, 18),
        maximum: new Date(2023, 7, 18),
        valueType: 'DateTime',
        labelFormat: 'MMM d',
        lineStyle: { width: 0 },
        majorGridLines: { width: 0 },
        edgeLabelPlacement: ej.base.Browser.isDevice ? 'None' : 'Shift',
        labelRotation: ej.base.Browser.isDevice ? -45 : 0,
        interval: ej.base.Browser.isDevice ? 2 : 1,
        crosshairTooltip: { enable: true }
    },
    primaryYAxis: {
        labelFormat: 'n2',
        majorTickLines: { width: 0 },
        lineStyle: { width: 0 },
        minimum: 0.86,
        maximum: 0.96,
        interval: 0.025
    },
    chartArea: { border: { width: 0 } },
    series: [
        {
            type: 'Spline', dataSource: synchronizedData, xName: 'USD',
            width: 2, yName: 'EUR', emptyPointSettings: { mode: 'Drop' }
        }
    ]
});

```

```

    ],
    chartMouseLeave: function (args) {
        chart1.hideCrosshair();
    },
    chartMouseMove: function (args) {
        if ((!ej.base.Browser.isDevice && !chart.isTouch &&
!chart.isChartDrag) || chart.startMove) {
            chart1.startMove = chart.startMove;
            chart1.showCrosshair(args.x, args.y);
        }
    },
    chartMouseUp: function (args) {
        if (ej.base.Browser.isDevice && chart.startMove) {
            chart1.hideCrosshair();
        }
    },
    title: 'US to EURO',
    titleStyle: { textAlignment: 'Near' },
    crosshair: { enable: true, lineType: 'Vertical', dashArray: '2,2' }
});
chart.appendTo('#container1');
charts.push(chart);
var chart1 = new ej.charts.Chart({
    primaryXAxis: {
        minimum: new Date(2023, 1, 18),
        maximum: new Date(2023, 7, 18),
        valueType: 'DateTime',
        labelFormat: 'MMM d',
        lineStyle: { width: 0 },
        majorGridLines: { width: 0 },
        edgeLabelPlacement: ej.base.Browser.isDevice ? 'None' : 'Shift',
        labelRotation: ej.base.Browser.isDevice ? -45 : 0,
        interval: ej.base.Browser.isDevice ? 2 : 1,
        crosshairTooltip: { enable: true }
    },
    primaryYAxis: {
        labelFormat: 'n1',
        majorTickLines: { width: 0 },
        lineStyle: { width: 0 },
        minimum: 79,
        maximum: 85,
        interval: 1.5
    },
    chartArea: { border: { width: 0 } },
    series: [
        {
            type: 'Area', dataSource: synchronizedData, xName: 'USD', width:
2, yName: 'INR', opacity: 0.6, border: { width: 2 }
        }
    ],
    chartMouseMove: function (args) {
        if ((!ej.base.Browser.isDevice && !chart1.isTouch &&
!chart1.isChartDrag) || chart1.startMove) {
            chart.startMove = chart1.startMove;
            chart.showCrosshair(args.x, args.y);
        }
    },
},

```

```

chartMouseLeave: function (args) {
    chart.hideCrosshair();
},
chartMouseUp: function (args) {
    if (ej.base.Browser.isDevice && chart1.startMove) {
        chart.hideCrosshair();
    }
},
title: 'US to INR',
titleStyle: { textAlignment: 'Near' },
crosshair: { enable: true, lineType: 'Vertical', dashArray: '2,2' },
});
chart1.appendTo('#container2');
charts.push(chart1);

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <script src="es5-datasource.js" type="text/javascript"></script>

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<style>
    #control-container {
        padding: 1px !important;
    }
    .row {
        display: flex;
    }
    .col {
        width: 50%;
        margin: 10px;
        height: 270px;
    }
</style>
<body>

    <div class="control-section">
        <div class="row">
            <div class="col" id="container1"></div>
            <div class="col" id="container2"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";

```

```

}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Zooming synchronization

You can maintain constant zoom levels across multiple charts using the [zoomComplete](#) event. In the [zoomComplete](#) event, obtain the [zoomFactor](#) and [zoomPosition](#) values of the particular chart, and then apply those values to the other charts.

INDEX.JS

```

var charts = [];
var zoomFactor = 0;
var zoomPosition = 0;
var chart = new ej.charts.Chart({
  primaryXAxis: {
    minimum: new Date(2023, 1, 18),
    maximum: new Date(2023, 7, 18),
    valueType: 'DateTime',
    labelFormat: 'MMM d',
    lineStyle: { width: 0 },
    majorGridLines: { width: 0 },
    edgeLabelPlacement: ej.base.Browser.isDevice ? 'None' : 'Shift',
    labelRotation: ej.base.Browser.isDevice ? -45 : 0,
    interval: ej.base.Browser.isDevice ? 2 : 1
  },
  primaryYAxis: {
    labelFormat: 'n2',
    majorTickLines: { width: 0 },
    lineStyle: { width: 0 },
    minimum: 0.86,
    maximum: 0.96,
    interval: 0.025
  },
  chartArea: { border: { width: 0 } },
  series: [
    {
      type: 'Line', dataSource: synchronizedData, xName: 'USD', width:
2, yName: 'EUR', emptyPointSettings: { mode: 'Drop' }
    }
  ],
  zoomSettings: {
    enableMouseWheelZooming: true,
    enablePinchZooming: true,
    enableScrollbar: false,
    enableDeferredZooming: false,
    enableSelectionZooming: true,
    enablePan: true,
    mode: 'X',
    toolbarItems: ['Pan', 'Reset']
  },
  zoomComplete: function (args) {
    if (args.axis.name === 'primaryXAxis') {
      zoomFactor = args.currentZoomFactor;
      zoomPosition = args.currentZoomPosition;
    }
  }
});

```

```

        zoomCompleteFunction(args);
    }
},
title: 'US to EURO',
titleStyle: { textAlign: 'Near' }
});
chart.appendTo('#container1');
charts.push(chart);
var chart1 = new ej.charts.Chart({
    primaryXAxis: {
        minimum: new Date(2023, 1, 18),
        maximum: new Date(2023, 7, 18),
        valueType: 'DateTime',
        labelFormat: 'MMM d',
        lineStyle: { width: 0 },
        majorGridLines: { width: 0 },
        edgeLabelPlacement: ej.base.Browser.isDevice ? 'None' : 'Shift',
        labelRotation: ej.base.Browser.isDevice ? -45 : 0,
        interval: ej.base.Browser.isDevice ? 2 : 1
    },
    primaryYAxis: {
        labelFormat: 'n1',
        majorTickLines: { width: 0 },
        lineStyle: { width: 0 },
        minimum: 79,
        maximum: 85,
        interval: 1.5
    },
    chartArea: { border: { width: 0 } },
    series: [
        {
            type: 'SplineArea', dataSource: synchronizedData, xName: 'USD',
            width: 2, yName: 'INR', opacity: 0.6, border: { width: 2 }
        }
    ],
    zoomSettings: {
        enableMouseWheelZooming: true,
        enablePinchZooming: true,
        enableScrollbar: false,
        enableDeferredZooming: false,
        enableSelectionZooming: true,
        enablePan: true,
        mode: 'X',
        toolbarItems: ['Pan', 'Reset']
    },
    zoomComplete: function (args) {
        if (args.axis.name === 'primaryXAxis') {
            zoomFactor = args.currentZoomFactor;
            zoomPosition = args.currentZoomPosition;
            zoomCompleteFunction(args);
        }
    },
    title: 'US to INR',
    titleStyle: { textAlign: 'Near' }
});
chart1.appendTo('#container2');
charts.push(chart1);

```



```
function zoomCompleteFunction(args) {
    for (var i = 0; i < charts.length; i++) {
        if (args.axis.series[0].chart.element.id !== charts[i].element.id) {
            charts[i].primaryXAxis.zoomFactor = zoomFactor;
            charts[i].primaryXAxis.zoomPosition = zoomPosition;
            charts[i].zoomModule.isZoomed =
args.axis.series[0].chart.zoomModule.isZoomed;
            charts[i].zoomModule.isPanning =
args.axis.series[0].chart.zoomModule.isPanning;
        }
    }
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <script src="es5-datasource.js" type="text/javascript"></script>

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<style>
    #control-container {
        padding: 1px !important;
    }
    .row {
        display: flex;
    }
    .col {
        width: 50%;
        margin: 10px;
        height: 270px;
    }
</style>
<body>

    <div class="control-section">
        <div class="row">
            <div class="col" id="container1"></div>
            <div class="col" id="container2"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
```

```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Selection synchronization

You can select the data across multiple charts using the [selectionComplete](#) event. In the `selectionComplete` event, obtain the selected values of the particular chart, and then apply those values to the other charts.

INDEX.JS

```
var charts = [];
var zoomFactor = 0;
var zoomPosition = 0;
var count = 0;
var chart = new ej.charts.Chart({
  primaryXAxis: {
    minimum: new Date(2023, 1, 18),
    maximum: new Date(2023, 7, 18),
    valueType: 'DateTime',
    labelFormat: 'MMM d',
    lineStyle: { width: 0 },
    majorGridLines: { width: 0 },
    edgeLabelPlacement: ej.base.Browser.isDevice ? 'None' : 'Shift',
    labelRotation: ej.base.Browser.isDevice ? -45 : 0,
    interval: ej.base.Browser.isDevice ? 2 : 1
  },
  primaryYAxis: {
    labelFormat: 'n2',
    majorTickLines: { width: 0 },
    lineStyle: { width: 0 },
    minimum: 0.86,
    maximum: 0.96,
    interval: 0.025
  },
  chartArea: { border: { width: 0 } },
  series: [
    {
      type: 'Line', dataSource: synchronizedData, xName: 'USD', width:
2, yName: 'EUR', emptyPointSettings: { mode: 'Drop' }
    }
  ],
  zoomSettings: {
    enableSelectionZooming: true,
    mode: 'X'
  },
  zoomComplete: function (args) {
    if (args.axis.name === 'primaryXAxis') {
      zoomFactor = args.currentZoomFactor;
      zoomPosition = args.currentZoomPosition;
      zoomCompleteFunction(args);
    }
  },
  selectionComplete: function (args) {
    selectionCompleteFunction(args);
  },
  selectionPattern: 'Box',
```

```

        selectionMode: 'Point',
        title: 'US to EURO',
        titleStyle: { textAlign: 'Near' }
    });
    chart.appendTo('#container1');
    charts.push(chart);
    var chart1 = new ej.charts.Chart({
        primaryXAxis: {
            minimum: new Date(2023, 1, 18),
            maximum: new Date(2023, 7, 18),
            valueType: 'DateTime',
            labelFormat: 'MMM d',
            lineStyle: { width: 0 },
            majorGridLines: { width: 0 },
            edgeLabelPlacement: ej.base.Browser.isDevice ? 'None' : 'Shift',
            labelRotation: ej.base.Browser.isDevice ? -45 : 0,
            interval: ej.base.Browser.isDevice ? 2 : 1
        },
        primaryYAxis: {
            labelFormat: 'n1',
            majorTickLines: { width: 0 },
            lineStyle: { width: 0 },
            minimum: 79,
            maximum: 85,
            interval: 1.5
        },
        chartArea: { border: { width: 0 } },
        series: [
            {
                type: 'Spline', dataSource: synchronizedData, xName: 'USD',
                width: 2, yName: 'INR', border: { width: 2 }
            }
        ],
        zoomSettings: {
            enableSelectionZooming: true,
            mode: 'X'
        },
        zoomComplete: function (args) {
            if (args.axis.name === 'primaryXAxis') {
                zoomFactor = args.currentZoomFactor;
                zoomPosition = args.currentZoomPosition;
                zoomCompleteFunction(args);
            }
        },
        selectionComplete: function (args) {
            selectionCompleteFunction(args);
        },
        selectionPattern: 'Box',
        selectionMode: 'Point',
        title: 'US to INR',
        titleStyle: { textAlign: 'Near' }
    });
    chart1.appendTo('#container2');
    charts.push(chart1);
    function zoomCompleteFunction(args) {
        for (var i = 0; i < charts.length; i++) {
            if (args.axis.series[0].chart.element.id !== charts[i].element.id) {

```

```

        charts[i].primaryXAxis.zoomFactor = zoomFactor;
        charts[i].primaryXAxis.zoomPosition = zoomPosition;
        charts[i].zoomModule.isZoomed =
args.axis.series[0].chart.zoomModule.isZoomed;
        charts[i].zoomModule.isPanning =
args.axis.series[0].chart.zoomModule.isPanning;
    }
}
}
function selectionCompleteFunction(args) {
    if (count == 0) {
        for (var j = 0; j < args.selectedDataValues.length; j++) {
            args.selectedDataValues[j].point =
args.selectedDataValues[j].pointIndex;
            args.selectedDataValues[j].series =
args.selectedDataValues[j].seriesIndex;
        }
        for (var i = 0; i < charts.length; i++) {
            if (args.chart.element.id !== charts[i].element.id) {
                charts[i].selectedDataIndexes = args.selectedDataValues;
                count += 1;
                charts[i].dataBind();
            }
        }
        count = 0;
    }
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <script src="es5-datasource.js" type="text/javascript"></script>

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<style>
    #control-container {
        padding: 1px !important;
    }
    .row {
        display: flex;
    }
    .col {
        width: 50%;
        margin: 10px;
        height: 270px;
    }
}

```

```

</style>
<body>

  <div class="control-section">
    <div class="row">
      <div class="col" id="container1"></div>
      <div class="col" id="container2"></div>
    </div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

<!-- markdownlint-disable MD036 -->

Selection in ##Platform_Name## Chart control

Chart provides selection support for the series and its data points on mouse click.

When Mouse is clicked on the data points, the corresponding series legend also will be selected.

We have different types of selection mode for selecting a data.

- None
- Point
- Series
- Cluster
- DragXY
- DragX
- DragY

Point

You can select a point, by setting `selectionMode` to point.

INDEX.JS

```

var chartData = [
  { country: "USA", gold: 50, silver: 70, bronze: 45 },
  { country: "China", gold: 40, silver: 60, bronze: 55 },
  { country: "Japan", gold: 70, silver: 60, bronze: 50 },
  { country: "Australia", gold: 60, silver: 56, bronze: 40 },
  { country: "France", gold: 50, silver: 45, bronze: 35 },
  { country: "Germany", gold: 40, silver: 30, bronze: 22 },
  { country: "Italy", gold: 40, silver: 35, bronze: 37 },
  { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    valueType: 'Category',
    title: 'Countries'
  }
});

```

```

    },
    primaryYAxis: {
        minimum: 0, maximum: 80,
        interval: 20, title: 'Medals'
    },
    series:[{
        dataSource: chartData,
        xName: 'country', yName: 'gold',
        name: 'Gold', type: 'Column'
    }, {
        dataSource: chartData,
        xName: 'country', yName: 'silver',
        name: 'Silver', type: 'Column'
    }, {
        dataSource: chartData,
        xName: 'country', yName: 'bronze',
        name: 'Bronze', type: 'Column'
    }],
    // selection mode as point
    selectionMode: 'Point',
    title: 'Olympic Medals'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script id="Unemployment" type="text/x-template">
    <div id='templateWrap'>
      <table style="width:100%; border: 1px solid black;">
        <tr><th colspan="2" bgcolor="#00FFFF">Unemployment</th></tr>
        <tr><td bgcolor="#00FFFF">${x}</td><td
bgcolor="#00FFFF">${y}</td></tr>
      </table>
    </div>
  </script>
<script>
var ele = document.getElementById('container');
if(ele) {

```

```

    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Series

You can select a series, by setting `selectionMode` to series.

INDEX.JS

```

var chartData = [
    { country: "USA", gold: 50, silver: 70, bronze: 45 },
    { country: "China", gold: 40, silver: 60, bronze: 55 },
    { country: "Japan", gold: 70, silver: 60, bronze: 50 },
    { country: "Australia", gold: 60, silver: 56, bronze: 40 },
    { country: "France", gold: 50, silver: 45, bronze: 35 },
    { country: "Germany", gold: 40, silver: 30, bronze: 22 },
    { country: "Italy", gold: 40, silver: 35, bronze: 37 },
    { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        valueType: 'Category',
        title: 'Countries'
    },
    primaryYAxis: {
        minimum: 0, maximum: 80,
        interval: 20, title: 'Medals'
    },
    series:[{
        dataSource: chartData,
        xName: 'country', yName: 'gold',
        name: 'Gold', type: 'Column'
    }, {
        dataSource: chartData,
        xName: 'country', yName: 'silver',
        name: 'Silver', type: 'Column'
    }, {
        dataSource: chartData,
        xName: 'country', yName: 'bronze',
        name: 'Bronze', type: 'Column'
    }],
    // selection mode as series
    selectionMode: 'Series',
    title: 'Olympic Medals'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">

```

```

<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
    <script id="Unemployment" type="text/x-template">
        <div id='templateWrap'>
            <table style="width:100%; border: 1px solid black;">
            <tr><th colspan="2" bgcolor="#00FFFF">Unemployment</th></tr>
            <tr><td bgcolor="#00FFFF">${x}</td><td
bgcolor="#00FFFF">${y}</td></tr>
            </table>
        </div>
    </script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Cluster

You can select the points that corresponds to the same index in all the series, by setting `selectionMode` to cluster.

INDEX.JS

```

var chartData = [
    { country: "USA", gold: 50, silver: 70, bronze: 45 },
    { country: "China", gold: 40, silver: 60, bronze: 55 },
    { country: "Japan", gold: 70, silver: 60, bronze: 50 },
    { country: "Australia", gold: 60, silver: 56, bronze: 40 },
    { country: "France", gold: 50, silver: 45, bronze: 35 },
    { country: "Germany", gold: 40, silver: 30, bronze: 22 },
    { country: "Italy", gold: 40, silver: 35, bronze: 37 },
    { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        valueType: 'Category',
        title: 'Countries'
    },
    primaryYAxis: {
        minimum: 0, maximum: 80,
        interval: 20, title: 'Medals'
    }
});

```



```

    },
    series:[{
        dataSource: chartData,
        xName: 'country', yName: 'gold',
        name: 'Gold', type: 'Column'
    }, {
        dataSource: chartData,
        xName: 'country', yName: 'silver',
        name: 'Silver', type: 'Column'
    }, {
        dataSource: chartData,
        xName: 'country', yName: 'bronze',
        name: 'Bronze', type: 'Column'
    }],
    // selection mode as cluster
    selectionMode: 'Cluster',
    title: 'Olympic Medals'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
    <script id="Unemployment" type="text/x-template">
        <div id='templateWrap'>
            <table style="width:100%; border: 1px solid black;">
                <tr><th colspan="2" bgcolor="#00FFFF">Unemployment</th></tr>
                <tr><td bgcolor="#00FFFF">${x}</td><td
bgcolor="#00FFFF">${y}</td></tr>
            </table>
        </div>
    </script>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
<script src="index.js" type="text/javascript"></script>

```

```
</body></html>
```

Rectangular selection

DragXY, DragX and DragY

To fetch a collection of data under a particular region, you have to set `selectionMode` as `DragXY`.

- DragXY - Allow us to select data with respect to both horizontal and vertical axis.
- DragX - Allow us to select data with respect to horizontal axis.
- DragY - Allow us to select data with respect to vertical axis.

The selected data's are returned as an array collection in the

[dragComplete](#) event.

INDEX.JS

```
var series1 = [];
var series2 = [];
var point1;
var value = 80;
var value1 = 70;
var i;
for (i = 1; i < 120; i++) {
    if (Math.random() > 0.5) {
        value += Math.random();
    } else {
        value -= Math.random();
    }
    value = value < 60 ? 60 : value > 90 ? 90 : value;
    point1 = { x: 120 + (i / 2), y: value.toFixed(1) };
    series1.push(point1);
}
for (i = 1; i < 120; i++) {
    if (Math.random() > 0.5) {
        value1 += Math.random();
    } else {
        value1 -= Math.random();
    }
    value1 = value1 < 60 ? 60 : value1 > 90 ? 90 : value1;
    point1 = { x: 120 + (i / 2), y: value1.toFixed(1) };
    series2.push(point1);
}
var chart = new ej.charts.Chart({
    primaryXAxis: {
        title: 'Height (cm)',
        minimum: 120, maximum: 180,
        edgeLabelPlacement: 'Shift',
        labelFormat: '{value}cm'
    },
    primaryYAxis: {
        title: 'Weight (kg)',
        minimum: 60, maximum: 90,
        labelFormat: '{value}kg',
```

```

        rangePadding: 'None'
    },
    series: [
        {
            type: 'Scatter',
            dataSource: series1, xName: 'x', yName: 'y',
            name: 'Male', opacity: 0.7,
            marker: { width: 10, height: 10 }
        }, {
            type: 'Scatter',
            dataSource: series2, xName: 'x', yName: 'y',
            name: 'Female', opacity: 0.7,
            marker: { width: 10, height: 10 }
        }
    ],
    // selection mode as dragxy
    selectionMode: 'DragXY',
    title: 'Height Vs Weight'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
    <script id="Unemployment" type="text/x-template">
        <div id='templateWrap'>
            <table style="width:100%; border: 1px solid black;">
                <tr><th colspan="2" bgcolor="#00FFFF">Unemployment</th></tr>
                <tr><td bgcolor="#00FFFF">${x}</td><td
bgcolor="#00FFFF">${y}</td></tr>
            </table>
        </div>
    </script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>

```

```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Selection type

You can select multiple points or series, by enabling the [isMultiSelect](#)

property.

INDEX.JS

```
var chartData = [
  { country: "USA", gold: 50, silver: 70, bronze: 45 },
  { country: "China", gold: 40, silver: 60, bronze: 55 },
  { country: "Japan", gold: 70, silver: 60, bronze: 50 },
  { country: "Australia", gold: 60, silver: 56, bronze: 40 },
  { country: "France", gold: 50, silver: 45, bronze: 35 },
  { country: "Germany", gold: 40, silver: 30, bronze: 22 },
  { country: "Italy", gold: 40, silver: 35, bronze: 37 },
  { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    valueType: 'Category',
    title: 'Countries'
  },
  primaryYAxis: {
    minimum: 0, maximum: 80,
    interval: 20, title: 'Medals'
  },
  series:[{
    dataSource: chartData,
    xName: 'country', yName: 'gold',
    name: 'Gold', type: 'Column'
  }, {
    dataSource: chartData,
    xName: 'country', yName: 'silver',
    name: 'Silver', type: 'Column'
  }, {
    dataSource: chartData,
    xName: 'country', yName: 'bronze',
    name: 'Bronze', type: 'Column'
  }],
  selectionMode: 'Point',
  // multipselect forselection
  isMultiSelect: true,
  title: 'Olympic Medals'
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
```

```

<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
    <script id="Unemployment" type="text/x-template">
        <div id='templateWrap'>
            <table style="width:100%; border: 1px solid black;">
                <tr><th colspan="2" bgcolor="#00FFFF">Unemployment</th></tr>
                <tr><td bgcolor="#00FFFF">${x}:</td><td
bgcolor="#00FFFF">${y}</td></tr>
            </table>
        </div>
    </script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Selection on load

You can able to select a point or series programmatically on a chart using [selectedDataIndexes](#) property.

INDEX.JS

```

var chartData = [
    { country: "USA", gold: 50, silver: 70, bronze: 45 },
    { country: "China", gold: 40, silver: 60, bronze: 55 },
    { country: "Japan", gold: 70, silver: 60, bronze: 50 },
    { country: "Australia", gold: 60, silver: 56, bronze: 40 },
    { country: "France", gold: 50, silver: 45, bronze: 35 },
    { country: "Germany", gold: 40, silver: 30, bronze: 22 },
    { country: "Italy", gold: 40, silver: 35, bronze: 37 },
    { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        valueType: 'Category',
        title: 'Countries'
    },
    primaryYAxis: {
        minimum: 0, maximum: 80,
        interval: 20, title: 'Medals'
    },
    series:[{

```

```

        dataSource: chartData,
        xName: 'country', yName: 'gold',
        name: 'Gold', type: 'Column',
        animation: {enable: false},
        selectionStyle: 'chartSelection1'
    }, {
        dataSource: chartData,
        xName: 'country', yName: 'silver',
        name: 'Silver', type: 'Column',
        animation: {enable: false},
        selectionStyle: 'chartSelection2'
    }, {
        dataSource: chartData,
        xName: 'country', yName: 'bronze',
        name: 'Bronze', type: 'Column',
        animation: {enable: false},
        selectionStyle: 'chartSelection3'
    }
  ],
  legendSettings: { visible: true, toggleVisibility: false,
enableHighlight: true},
  // Selcted data indexes for chart series
  title: 'Olympic Medals'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script id="Unemployment" type="text/x-template">
    <div id='templateWrap'>
      <table style="width:100%; border: 1px solid black;">
        <tr><th colspan="2" bgcolor="#00FFFF">Unemployment</th></tr>
        <tr><td bgcolor="#00FFFF">${x}</td><td
bgcolor="#00FFFF">${y}</td></tr>
      </table>
    </div>
  </script>
  <script>
var ele = document.getElementById('container');

```

```

if(ele) {
    ele.style.visibility = "visible";
}

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Selection through on legend

You can able to select a point or series through on legend using [toggleVisibility](#) property. Also, use [enableHighlight](#) property for highlighting the series through legend.

INDEX.JS

```

var chartData = [
    { country: "USA", gold: 50, silver: 70, bronze: 45 },
    { country: "China", gold: 40, silver: 60, bronze: 55 },
    { country: "Japan", gold: 70, silver: 60, bronze: 50 },
    { country: "Australia", gold: 60, silver: 56, bronze: 40 },
    { country: "France", gold: 50, silver: 45, bronze: 35 },
    { country: "Germany", gold: 40, silver: 30, bronze: 22 },
    { country: "Italy", gold: 40, silver: 35, bronze: 37 },
    { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        valueType: 'Category',
        title: 'Countries'
    },
    primaryYAxis: {
        minimum: 0, maximum: 80,
        interval: 20, title: 'Medals'
    },
    series:[{
        dataSource: chartData,
        xName: 'country', yName: 'gold',
        name: 'Gold', type: 'Column',
        animation: {enable: false},
        selectionStyle: 'chartSelection1'
    }, {
        dataSource: chartData,
        xName: 'country', yName: 'silver',
        name: 'Silver', type: 'Column',
        animation: {enable: false},
        selectionStyle: 'chartSelection2'
    }, {
        dataSource: chartData,
        xName: 'country', yName: 'bronze',
        name: 'Bronze', type: 'Column',
        animation: {enable: false},
        selectionStyle: 'chartSelection3'
    }],
    legendSettings: { visible: true, toggleVisibility: false,
enableHighlight: true},
    // Selcted data indexes for chart series
    title: 'Olympic Medals'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script id="Unemployment" type="text/x-template">
    <div id='templateWrap'>
      <table style="width:100%; border: 1px solid black;">
        <tr><th colspan="2" bgcolor="#00FFFF">Unemployment</th></tr>
        <tr><td bgcolor="#00FFFF">${x}</td><td
bgcolor="#00FFFF">${y}</td></tr>
      </table>
    </div>
  </script>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Customization for selection

You can apply custom style to selected points or series with [selectionStyle](#) property.

INDEX.JS

```

var chartData = [
  { country: "USA", gold: 50, silver: 70, bronze: 45 },
  { country: "China", gold: 40, silver: 60, bronze: 55 },
  { country: "Japan", gold: 70, silver: 60, bronze: 50 },
  { country: "Australia", gold: 60, silver: 56, bronze: 40 },
  { country: "France", gold: 50, silver: 45, bronze: 35 },
  { country: "Germany", gold: 40, silver: 30, bronze: 22 },
  { country: "Italy", gold: 40, silver: 35, bronze: 37 },
  { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart = new ej.charts.Chart({

```



```

primaryXAxis: {
    valueType: 'Category',
    title: 'Countries'
},
primaryYAxis: {
    minimum: 0, maximum: 80,
    interval: 20, title: 'Medals'
},
series:[{
    dataSource: chartData,
    xName: 'country', yName: 'gold',
    name: 'Gold', type: 'Column',
    //Selection style for chart series selection
    selectionStyle: 'chartSelection1'
}, {
    dataSource: chartData,
    xName: 'country', yName: 'silver',
    name: 'Silver', type: 'Column',
    selectionStyle: 'chartSelection2'
}, {
    dataSource: chartData,
    xName: 'country', yName: 'bronze',
    name: 'Bronze', type: 'Column',
    selectionStyle: 'chartSelection3'
}],
selectionMode: 'Point',
isMultiSelect: true,
title: 'Olympic Medals'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script id="Unemployment" type="text/x-template">
    <div id='templateWrap'>
      <table style="width:100%; border: 1px solid black;">
        <tr><th colspan="2" bgcolor="#00FFFF">Unemployment</th></tr>

```

```

        <tr><td bgcolor="#00FFFF">${x}</td><td
        bgcolor="#00FFFF">${y}</td></tr>
    </table>
</div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Display selected data for range selection](#)

Chart print in ##Platform_Name## Chart control

Print

The rendered chart can be printed directly from the browser by calling the public method print. ID of the chart div element must be passed as argument to that method.

INDEX.JS

```

var chart = new ej.charts.Chart({
    //Initializing Primary X Axis
    primaryXAxis: {
        title: 'Manager',
        valueType: 'Category',
        majorGridLines: { width: 0 }
    },
    //Initializing Primary Y Axis
    primaryYAxis: {
        title: 'Sales',
        minimum: 0,
        maximum: 20000,
        majorGridLines: { width: 0 }
    },
    //Initializing Chart Series
    series: [
        {
            type: 'Column',
            dataSource: [{ x: 'John', y: 10000 }, { x: 'Jake', y: 12000 }, { x: 'Peter', y: 18000 }, { x: 'James', y: 11000 }, { x: 'Mary', y: 9700 }],
            xName: 'x', width: 2,
            yName: 'y'
        }
    ],
    //Initializing Chart title
    title: 'Sales Comparision',
    '#element');

```

```
document.getElementById('print').onclick = () => {
    chart.print();
};
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element" style="float: left "></div>
    <button id="print" type="button" width="15%" style="float:
right">Print</button>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Export

The rendered chart can be exported to JPEG, PNG, SVG, PDF, XLSX, or CSV format using the export method in chart. The input parameters for this method are type for format and fileName for result.

The optional parameters for this method are,

- orientation - either portrait or landscape mode during PDF export,
- controls - pass collections of controls for multiple export,
- width - width of chart export, and
- height - height of chart export.

INDEX.JS

```
var chart = new ej.charts.Chart({
  //Initializing Primary X Axis
  primaryXAxis: {
    title: 'Manager',
```

```

        valueType: 'Category',
        majorGridLines: { width: 0 }
    },
    //Initializing Primary Y Axis
    primaryYAxis:
    {
        title: 'Sales',
        minimum: 0,
        maximum: 20000,
        majorGridLines: { width: 0 }
    },
    //Initializing Chart Series
    series: [
        {
            type: 'Column',
            dataSource: [{ x: 'John', y: 10000 }, { x: 'Jake', y: 12000
}, { x: 'Peter', y: 18000 },
            { x: 'James', y: 11000 }, { x: 'Mary', y: 9700 }],
            xName: 'x', width: 2,
            yName: 'y'
        }
    ],
    //Initializing Chart title
    title: 'Sales Comparision',
}, '#element');
document.getElementById('print').onclick = () => {
    chart.export('PNG', 'result');
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element" style="float: left "></div>
        <div id="element1" style="float: left "></div>
        <button id="print" type="button" width="15%" style="float:
right">Export</button>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {

```

```

    ele.style.visibility = "visible";
  }
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Adding header and footer in PDF export

In the export method, specify the following parameters to add a header and footer text to the exported PDF document:

- **header** - Specify the text that should appear at the top of the exported PDF document.
- **footer** - Specify the text that should appear at the bottom of the exported PDF document.

INDEX.JS

```

var chart = new ej.charts.Chart({
  //Initializing Primary X Axis
  primaryXAxis: {
    title: 'Manager',
    valueType: 'Category',
    majorGridLines: { width: 0 }
  },
  //Initializing Primary Y Axis
  primaryYAxis: {
    title: 'Sales',
    minimum: 0,
    maximum: 20000,
    majorGridLines: { width: 0 }
  },
  //Initializing Chart Series
  series: [
    {
      type: 'Column',
      dataSource: [{ x: 'John', y: 10000 }, { x: 'Jake', y: 12000 }, {
x: 'Peter', y: 18000 },
      { x: 'James', y: 11000 }, { x: 'Mary', y: 9700 }],
      xName: 'x', width: 2,
      yName: 'y'
    }
  ],
  //Initializing Chart title
  title: 'Sales Comparision',
}, '#element');
document.getElementById('export').onclick = () => {
  const header = {
    content: 'Chart Header',
    fontSize: 15
  };
  const footer = {
    content: 'Chart Footer',
    fontSize: 15,
  };
};

```

```
chart.exportModule.export('PDF', 'Chart', 1, [chart], null, null, true,
header, footer);
};
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element" style="float: left "></div>
    <div id="element1" style="float: left "></div>
    <button id="export" type="button" width="15%" style="float:
right">Export</button>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Exporting charts into separate page during the PDF export

During PDF export, set the `exportToMultiplePage` parameter to **true** to export each chart as a separate page.

INDEX.JS

```
var chart = new ej.charts.Chart({
  primaryXAxis: {
    valueType: 'DateTime',
    labelFormat: 'y',
    intervalType: 'Years',
    edgeLabelPlacement: 'Shift',
    majorGridLines: { width: 0 }
  },
  //Initializing Primary Y Axis
  primaryYAxis:
  {
    labelFormat: '{value}%',
```

```

        rangePadding: 'None',
        minimum: 0,
        maximum: 100,
        interval: 20,
        lineStyle: { width: 0 },
        majorTickLines: { width: 0 },
        minorTickLines: { width: 0 }
    },
    //Initializing Chart Series
    series: [
        {
            type: 'Line',
            dataSource: [{ x: new Date(2005, 0, 1), y: 21 }, { x: new
Date(2006, 0, 1), y: 24 },
            { x: new Date(2007, 0, 1), y: 36 }, { x: new Date(2008, 0, 1),
y: 38 },
            { x: new Date(2009, 0, 1), y: 54 }, { x: new Date(2010, 0, 1),
y: 57 },
            { x: new Date(2011, 0, 1), y: 70 }
            ],
            xName: 'x', width: 2, yName: 'y', name: 'Germany',
            marker: { visible: true, width: 10, height: 10 },
        },
        {
            type: 'Line',
            dataSource: [{ x: new Date(2005, 0, 1), y: 28 }, { x: new
Date(2006, 0, 1), y: 44 },
            { x: new Date(2007, 0, 1), y: 48 }, { x: new Date(2008, 0, 1),
y: 50 },
            { x: new Date(2009, 0, 1), y: 66 }, { x: new Date(2010, 0, 1),
y: 78 },
            { x: new Date(2011, 0, 1), y: 84 }
            ],
            xName: 'x', width: 2, yName: 'y', name: 'England',
            marker: { visible: true, width: 10, height: 10 },
        }
    ],
    title: 'Medal Count',
}, '#element');
var chart1 = new ej.charts.Chart({
    primaryXAxis: {
        title: 'Manager',
        valueType: 'Category',
        majorGridLines: { width: 0 }
    },
    primaryYAxis:
    {
        title: 'Sales',
        minimum: 0,
        maximum: 20000,
        majorGridLines: { width: 0 }
    },
    series: [
        {
            type: 'Column',
            dataSource: [{ x: 'John', y: 10000 }, { x: 'Jake', y: 12000 }, {
x: 'Peter', y: 18000 },

```

```

        { x: 'James', y: 11000 }, { x: 'Mary', y: 9700 }]],
        xName: 'x', width: 2,
        yName: 'y'
    }
},
title: 'Sales Comparision',
}, '#element1');
var pie = new ej.charts.AccumulationChart({
    series: [
        {
            dataSource: [{ x: 'Labour', y: 18, text: '18%' }, { x: 'Legal',
y: 8, text: '8%' },
            { x: 'Production', y: 15, text: '15%' }, { x: 'License', y: 11,
text: '11%' },
            { x: 'Facilities', y: 18, text: '18%' }, { x: 'Taxes', y: 14,
text: '14%' },
            { x: 'Insurance', y: 16, text: '16%' }]],
            dataLabel: {
                visible: true,
                name: 'text',
                position: 'Inside',
                font: {
                    fontWeight: '600',
                    color: 'ffffff'
                }
            },
            radius: '70%', xName: 'x',
            yName: 'y', startAngle: 0,
            endAngle: 360,
            name: 'Project'
        }
    ],
    enableSmartLabels: true,
    legendSettings: {
        visible: true
    },
    tooltip: { enable: false },
    title: 'Project Cost Breakdown'
}, '#element2');
document.getElementById('print').onclick = () => {
    chart.exportModule.export('PDF', 'Chart', null, [chart, chart1, pie],
null, null, true, undefined, undefined, true);
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

```



```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id='container'>
        <div id='element' style="float: left "></div>
        <div id='element1' style="float: left "></div>
        <div id='element2' style="float: left "></div>
        <button id= "print" type="button" width ='15%' style="float:
right">Export</button>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Multiple chart export

You can export the multiple charts in single page by passing the multiple chart objects in the export method of chart. To export multiple charts in a single page, follow the given steps:

Initially, render more than one chart to export, and then add button to export the multiple charts. In button click, call the export method in charts, and then pass the multiple chart objects in the export method.

INDEX.JS

```

var chart = new ej.charts.Chart({
    //Initializing Primary X Axis
    primaryXAxis: {
        title: 'Manager',
        valueType: 'Category',
        majorGridLines: { width: 0 }
    },
    //Initializing Primary Y Axis
    primaryYAxis: {
        title: 'Sales',
        minimum: 0,
        maximum: 20000,
        majorGridLines: { width: 0 }
    },
    //Initializing Chart Series
    series: [
        {
            type: 'Column',
            dataSource: [{ x: 'John', y: 10000 }, { x: 'Jake', y: 12000 }, {
x: 'Peter', y: 18000 },
                { x: 'James', y: 11000 }, { x: 'Mary', y: 9700 }],
            xName: 'x', width: 2,
            yName: 'y'
        }
    ]
});

```

```

    },
    //Initializing Chart title
    title: 'Sales Comparision',
  }, '#element');
var chart1 = new ej.charts.Chart({
  //Initializing Primary X Axis
  primaryXAxis: {
    title: 'Manager',
    valueType: 'Category',
    majorGridLines: { width: 0 }
  },
  //Initializing Primary Y Axis
  primaryYAxis: {
    title: 'Sales',
    minimum: 0,
    maximum: 20000,
    majorGridLines: { width: 0 }
  },
  //Initializing Chart Series
  series: [
    {
      type: 'Column',
      dataSource: [{ x: 'John', y: 10000 }, { x: 'Jake', y: 12000 }, {
x: 'Peter', y: 18000 },
      { x: 'James', y: 11000 }, { x: 'Mary', y: 9700 }],
      xName: 'x', width: 2,
      yName: 'y'
    }
  ],
  //Initializing Chart title
  title: 'Sales Comparision',
}, '#element1');
document.getElementById('print').onclick = function () {
  chart.export('PNG', Chart, null, [chart, chart1]);
};
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```

```

<div id="container">
  <div id="element" style="float: left "></div>
  <div id="element1" style="float: left "></div>
  <button id="print" type="button" width="15%" style="float:
right">Export</button>
</div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Exporting chart using base64 string

The chart can be exported as an image in the form of a base64 string by utilizing HTML canvas. This process involves rendering the chart onto a canvas element and then converting the canvas content to a base64 string.

INDEX.JS

```

var chart = new ej.charts.Chart({
  //Initializing Primary X Axis
  primaryXAxis: {
    valueType: 'Category',
    majorGridLines: { width: 0 },
    majorTickLines: { width: 0 },
    minorTickLines: { width: 0 }
  },
  //Initializing Primary Y Axis
  primaryYAxis: {
    {
      minimum: 0,
      maximum: 40,
      interval: 10,
      lineStyle: {width : 0},
      minorTickLines: {width: 0},
      majorTickLines: {width : 0},
    },
    chartArea: {
      border: {
        width: 0
      }
    }
  },
  //Initializing Chart Series
  series: [
    {
      type: 'Column',
      dataSource: [
        { x: 'DEU', y: 35.5 }, { x: 'CHN', y: 18.3 }, { x: 'ITA', y:
17.6 }, { x: 'JPN', y: 13.6 },
        { x: 'US', y: 12 }, { x: 'ESP', y: 5.6 }, { x: 'FRA', y: 4.6
}, { x: 'AUS', y: 3.3 },
        { x: 'BEL', y: 3 }, { x: 'UK', y: 2.9 }
      ]
    }
  ]
}

```

```

        xName: 'x', width: 2,
        yName: 'y'
    }
},
],
}, '#element');
document.getElementById('export').onclick = function () {
    var svg = document.querySelector("#element_svg");
    var svgData = new XMLSerializer().serializeToString(svg);
    var canvas = document.createElement("canvas");
    document.body.appendChild(canvas);
    var svgSize = svg.getBoundingClientRect();
    canvas.width = svgSize.width;
    canvas.height = svgSize.height;
    var ctx = canvas.getContext("2d");
    var img = document.createElement("img");
    img.setAttribute("src", "data:image/svg+xml;base64," +
btoa(svgData));
    img.onload = function() {
        ctx.drawImage(img, 0, 0);
        var imagedata = canvas.toDataURL("image/png");
        console.log(imagedata); // printed base64 in console
        canvas.remove();
    };
};
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element" style="float: left "></div>
        <button id="export" type="button" width="15%" style="float:
right">Export</button>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Chart appearance in ##Platform_Name## Chart control

Custom color palette

You can customize the default color of series or points by providing a custom color palette of your choice by using the [palettes](#) property.

INDEX.JS

```
var chartData = [
  { country: "USA", gold: 50, silver: 70, bronze: 45 },
  { country: "China", gold: 40, silver: 60, bronze: 55 },
  { country: "Japan", gold: 70, silver: 60, bronze: 50 },
  { country: "Australia", gold: 60, silver: 56, bronze: 40 },
  { country: "France", gold: 50, silver: 45, bronze: 35 },
  { country: "Germany", gold: 40, silver: 30, bronze: 22 },
  { country: "Italy", gold: 40, silver: 35, bronze: 37 },
  { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    valueType: 'Category',
    title: 'Countries'
  },
  primaryYAxis: {
    minimum: 0, maximum: 80,
    interval: 20, title: 'Medals'
  },
  series:[{
    dataSource: chartData,
    xName: 'country', yName: 'gold',
    name: 'Gold', type: 'Column'
  }, {
    dataSource: chartData,
    xName: 'country', yName: 'silver',
    name: 'Silver', type: 'Column'
  }, {
    dataSource: chartData,
    xName: 'country', yName: 'bronze',
    name: 'Bronze', type: 'Column'
  }],
  // palettes for chart
  palettes: ["#E94649", "#F6B53F", "#6FAAB0", "#C4C24A"],
  title: 'Olympic Medals'
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
        <div id="element1"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Data point customization

The color of individual data point or data points within a range can be customized using the options below.

Point color mapping

You can bind the color for the points from [dataSource](#) for the series using [pointColorMapping](#) property.

INDEX.JS

```

var chart = new ej.charts.Chart({
    primaryXAxis: { valueType: 'Category', majorGridLines: { width: 0 } },
    primaryYAxis: {
        lineStyle: { width: 0 },
        majorTickLines: { width: 0 },
        minorTickLines: { width: 0 },
        labelFormat: '{value}°C',
    },
    chartArea: {
        border: {
            width: 0
        }
    },
    series: [
        {
            pointColorMapping: "color",
            dataSource: [
                { x: 'Jan', y: 6.96, color: "red" },
                { x: 'Feb', y: 8.9, color: "blue" },
                { x: 'Mar', y: 12, color: "orange" },
                { x: 'Apr', y: 17.5, color: "aqua" },
                { x: 'May', y: 22.1, color: "grey" }
            ], xName: 'x', yName: 'y', type: 'Column',
            animation: { enable: false },
            cornerRadius: {
                topLeft: 10, topRight: 10
            }
        }
    ]
});

```

```

        },
    ],
    title: 'USA CLIMATE - WEATHER BY MONTH',
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
    <div id="element1"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Range color mapping

You can differentiate data points based on their y values using [rangeColorSettings](#) in the chart.

INDEX.JS

```

var chart = new ej.charts.Chart({
  selectionMode: 'Point',
  primaryXAxis: { valueType: 'Category', majorGridLines: { width: 0 }
},
  primaryYAxis: {
    lineStyle: { width: 0 },
    majorTickLines: { width: 0 },
    minorTickLines: { width: 0 },
    labelFormat: '{value}°C',
  },
  chartArea: {
    border: {
      width: 0
    }
  }
});

```

```

    },
    series: [
        {
            dataSource: [
                { x: 'Jan', y: 6.96 },
                { x: 'Feb', y: 8.9 },
                { x: 'Mar', y: 12 },
                { x: 'Apr', y: 17.5 },
                { x: 'May', y: 22.1 },
                { x: 'June', y: 25 },
                { x: 'July', y: 29.4 },
                { x: 'Aug', y: 29.6 },
                { x: 'Sep', y: 25.8 },
                { x: 'Oct', y: 21.1 },
                { x: 'Nov', y: 15.5 },
                { x: 'Dec', y: 9.9 }
            ], xName: 'x', yName: 'y', type: 'Column',
            animation: { enable: false }, name: 'USA',
            cornerRadius: {
                topLeft: 10, topRight: 10
            },
        }
    ],
    rangeColorSettings: [
        {
            label: '1°C to 10°C',
            start: 1,
            end: 10,
            colors: ['#F9D422']
        },
        {
            label: '11°C to 20°C',
            start: 11,
            end: 20,
            colors: ['#F28F3F']
        },
        {
            label: '21°C to 30°C',
            start: 21,
            end: 30,
            colors: ['#E94F53']
        }
    ],
    legendSettings: {
        mode: 'Range',
        toggleVisibility: false
    },
    title: 'USA CLIMATE - WEATHER BY MONTH',
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">

```



```

<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
            <div id="element1"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Point level customization

Marker, data label and fill color of each data point can be customized with [pointRender](#) and [textRender](#) event.

INDEX.JS

```

var chartData = [
    { country: "USA", gold: 50 },
    { country: "China", gold: 40 },
    { country: "Japan", gold: 70 },
    { country: "Australia", gold: 60 },
    { country: "France", gold: 50 },
    { country: "Germany", gold: 40 },
    { country: "Italy", gold: 40 },
    { country: "Sweden", gold: 30 }
];
var colors = ['#00bdae', '#404041', '#357cd2', '#e56590', '#f8b883',
    '#70ad47', '#dd8abd', '#7f84e8', '#7bb4eb', '#ea7a57'];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        valueType: 'Category',
        title: 'Countries'
    },
    primaryYAxis: {
        minimum: 0, maximum: 80,
        interval: 20, title: 'Medals'
    },
    series: [{
        dataSource: chartData,
        xName: 'country', yName: 'gold',
        name: 'Gold', type: 'Column'
    }]
});

```

```

    }],
    // point render event for chart
    pointRender: (args) => {
        args.fill = colors[args.point.index];
    },
    title: 'Olympic Medals'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
    <div id="element1"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

<!-- markdownlint-disable MD036 -->

Chart area customization

<!-- markdownlint-disable MD036 -->

Customize the chart background

Using [background](#) and [border](#) properties, you can change the background color and border of the chart.

INDEX.JS

```

var chartData = [
  { country: "USA", gold: 50 },
  { country: "China", gold: 40 },
  { country: "Japan", gold: 70 },
  { country: "Australia", gold: 60 },
  { country: "France", gold: 50 },

```

```

    { country: "Germany", gold: 40 },
    { country: "Italy", gold: 40 },
    { country: "Sweden", gold: 30 }
  ];
  var chart = new ej.charts.Chart({
    primaryXAxis: {
      valueType: 'Category',
      title: 'Countries'
    },
    primaryYAxis: {
      minimum: 0, maximum: 80,
      interval: 20, title: 'Medals'
    },
    series: [{
      dataSource: chartData,
      xName: 'country', yName: 'gold',
      name: 'Gold', type: 'Column',
      border: { width: 2, color: 'grey' }
    }],
    title: 'Olympic Medals',
    //Customizing Chart background
    background: 'skyblue',
    //Customize the chart border and opacity
    border: { color: "#FF0000", width: 2 },
  }, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
    <div id="element1"></div>
  </div>
  <script>
  var ele = document.getElementById('container');
  if(ele) {
    ele.style.visibility = "visible";
  }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Chart margin

You can set margin for chart from its container through [margin](#) property.

INDEX.JS

```
var chartData = [
  { country: "USA", gold: 50 },
  { country: "China", gold: 40 },
  { country: "Japan", gold: 70 },
  { country: "Australia", gold: 60 },
  { country: "France", gold: 50 },
  { country: "Germany", gold: 40 },
  { country: "Italy", gold: 40 },
  { country: "Sweden", gold: 30 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    valueType: 'Category',
    title: 'Countries'
  },
  primaryYAxis: {
    minimum: 0, maximum: 80,
    interval: 20, title: 'Medals'
  },
  series: [{
    dataSource: chartData,
    xName: 'country', yName: 'gold',
    name: 'Gold', type: 'Column',
    border: { width: 2, color: 'grey' }
  }],
  title: 'Olympic Medals',
  background: 'skyblue',
  border: { color: "#FF0000", width: 2 },
  //Change chart margin to left, right, top and bottom
  margin: { left: 40, right: 40, top: 40, bottom: 40 },
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>
```

```

    <div id="container">
        <div id="element"></div>
        <div id="element1"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Chart area customization

Using [background](#) and [border](#) properties, you can change the background color and border of the chart area. Width for the chart area can be customized using [width](#) property.

INDEX.JS

```

var chartData = [
    { country: "USA", gold: 50 },
    { country: "China", gold: 40 },
    { country: "Japan", gold: 70 },
    { country: "Australia", gold: 60 },
    { country: "France", gold: 50 },
    { country: "Germany", gold: 40 },
    { country: "Italy", gold: 40 },
    { country: "Sweden", gold: 30 }
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        valueType: 'Category',
        title: 'Countries'
    },
    primaryYAxis: {
        minimum: 0, maximum: 80,
        interval: 20, title: 'Medals'
    },
    series: [{
        dataSource: chartData,
        xName: 'country', yName: 'gold',
        name: 'Gold', type: 'Column',
        border: { width: 2, color: 'grey' }
    }],
    title: 'Olympic Medals',
    chartArea: {
        //background for Chart area
        background: "skyblue",
        // width of the chart area
        width: '80%'
    }
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
    <div id="element1"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Animation

You can customize animation for a particular series using [animation](#) property. You can enable or disable animation of the series using [enable](#) property, [duration](#) specifies the duration of an animation and [delay](#) allows us to start the animation at desire time.

INDEX.JS

```

var chartData = [
  { country: "USA", gold: 50 },
  { country: "China", gold: 40 },
  { country: "Japan", gold: 70 },
  { country: "Australia", gold: 60 },
  { country: "France", gold: 50 },
  { country: "Germany", gold: 40 },
  { country: "Italy", gold: 40 },
  { country: "Sweden", gold: 30 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    valueType: 'Category',
    title: 'Countries'
  },
  primaryYAxis: {
    minimum: 0, maximum: 80,
    interval: 20, title: 'Medals'
  },
  series:[{

```

```

        dataSource: chartData,
        xName: 'country', yName: 'gold',
        name: 'Gold', type: 'Column',
        border:{ width: 2, color: 'grey'},
        //Animation for chart series
        animation:{
            enable: true,
            duration: 2000,
            delay: 200
        }
    }],
    title: 'Olympic Medals'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
    <div id="element1"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Fluid animation

Fluid animation used to animate series with updated dataSource continues animation rather than animation whole series. You can customize animation for a particular series using `[animate]` method.

INDEX.JS

```

var count = 0;
var chart = new ej.charts.Chart({
  //Initializing Primary X Axis
  primaryXAxis: {

```

```

        valueType: 'Category', interval: 1,
        tickPosition: 'Inside',
        labelPosition: 'Inside', labelStyle: { color: '#ffffff' }
    },
    chartArea: { border: { width: 0 } },
    //Initializing Primary Y Axis
    primaryYAxis: {
        minimum: 0, maximum: 300, interval: 50,
        labelStyle: { color: 'transparent' }
    },
    //Initializing Chart Series
    series: [
        {
            type: 'Column', xName: 'x', width: 2, yName: 'y',
            dataSource: [
                { x: 'Egg', y: 106, text: 'Bangaladesh' },
                { x: 'Fish', y: 103, text: 'Bhutn' },
                { x: 'Misc', y: 198, text: 'Nepal' },
                { x: 'Tea', y: 189, text: 'Thiland' },
                { x: 'Fruit', y: 250, text: 'Malaysia' }
            ], name: 'Tiger',
            cornerRadius: {
                bottomLeft: 10, bottomRight: 10, topLeft: 10, topRight:
10
            },
        },
    ],
    legendSettings: { visible: false },
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
        <div id="element1"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";

```



```

}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Chart title

Chart can be given a title using [title](#) property, to show the information about the data plotted.

INDEX.JS

```

var chartData = [
  { x: new Date(1975, 0, 1), y: 16, y1: 10, y2: 4.5 },
  { x: new Date(1980, 0, 1), y: 12.5, y1: 7.5, y2: 5 },
  { x: new Date(1985, 0, 1), y: 19, y1: 11, y2: 6.5 },
  { x: new Date(1990, 0, 1), y: 14.4, y1: 7, y2: 4.4 },
  { x: new Date(1995, 0, 1), y: 11.5, y1: 8, y2: 5 },
  { x: new Date(2000, 0, 1), y: 14, y1: 6, y2: 1.5 },
  { x: new Date(2005, 0, 1), y: 10, y1: 3.5, y2: 2.5 },
  { x: new Date(2010, 0, 1), y: 16, y1: 7, y2: 3.7 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    title: 'Years',
    lineStyle: { width: 0 },
    labelFormat: 'y',
    intervalType: 'Years',
    valueType: 'DateTime',
    edgeLabelPlacement: 'Shift'
  },
  primaryYAxis: {
    {
      title: 'Percentage (%)',
      minimum: 0, maximum: 20, interval: 2,
      labelFormat: '{value}%'
    }
  },
  series: [
    {
      type: 'StepLine',
      dataSource: chartData, xName: 'x', yName: 'y',
      width: 2, name: 'China',
      marker: {
        visible: true, width: 10, height: 10
      }
    },
    {
      type: 'StepLine',
      dataSource: chartData, xName: 'x', yName: 'y1',
      width: 2, name: 'Australia',
      marker: {
        visible: true, width: 10, height: 10
      }
    },
    {
      type: 'StepLine',
      dataSource: chartData, xName: 'x', yName: 'y2',
      width: 2, name: 'Japan',

```

```

        marker: {
            visible: true, width: 10, height: 10
        },
    ],
    //Title for chart
    title: 'Unemployment Rates 1975-2010',
    titleStyle: {
        fontFamily: 'Arial',
        fontStyle: 'italic',
        fontWeight: 'regular',
        color: '#E27F2D',
        size: '23px'
    }
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
    <div id="element1"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Title position

By using the [position](#) property in [titleStyle](#), you can position the [title](#) at left, right, top or bottom of the chart. The title is positioned at the top of the chart, by default.

INDEX.JS

```

var chartData = [
  { x: new Date(1975, 0, 1), y: 16, y1: 10, y2: 4.5 },

```

```

    { x: new Date(1980, 0, 1), y: 12.5, y1: 7.5, y2: 5 },
    { x: new Date(1985, 0, 1), y: 19, y1: 11, y2: 6.5 },
    { x: new Date(1990, 0, 1), y: 14.4, y1: 7, y2: 4.4 },
    { x: new Date(1995, 0, 1), y: 11.5, y1: 8, y2: 5 },
    { x: new Date(2000, 0, 1), y: 14, y1: 6, y2: 1.5 },
    { x: new Date(2005, 0, 1), y: 10, y1: 3.5, y2: 2.5 },
    { x: new Date(2010, 0, 1), y: 16, y1: 7, y2: 3.7 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    title: 'Years',
    lineStyle: { width: 0 },
    labelFormat: 'y',
    intervalType: 'Years',
    valueType: 'DateTime',
    edgeLabelPlacement: 'Shift'
  },
  primaryYAxis: {
    {
      title: 'Percentage (%)',
      minimum: 0, maximum: 20, interval: 2,
      labelFormat: '{value}%'
    },
  },
  series: [
    {
      type: 'StepLine',
      dataSource: chartData, xName: 'x', yName: 'y',
      width: 2, name: 'China',
      marker: {
        visible: true, width: 10, height: 10
      },
    },
    {
      type: 'StepLine',
      dataSource: chartData, xName: 'x', yName: 'y1',
      width: 2, name: 'Australia',
      marker: {
        visible: true, width: 10, height: 10
      },
    },
    {
      type: 'StepLine',
      dataSource: chartData, xName: 'x', yName: 'y2',
      width: 2, name: 'Japan',
      marker: {
        visible: true, width: 10, height: 10
      },
    },
  ],
  //Title for chart
  title: 'Unemployment Rates 1975-2010',
  titleStyle: { position: 'Bottom' },
  legendSettings: { visible: false }
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
    <div id="element1"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

The custom option helps you to position the title anywhere in the chart using [x](#) and [y](#) coordinates.

INDEX.JS

```

var chartData = [
  { x: new Date(1975, 0, 1), y: 16, y1: 10, y2: 4.5 },
  { x: new Date(1980, 0, 1), y: 12.5, y1: 7.5, y2: 5 },
  { x: new Date(1985, 0, 1), y: 19, y1: 11, y2: 6.5 },
  { x: new Date(1990, 0, 1), y: 14.4, y1: 7, y2: 4.4 },
  { x: new Date(1995, 0, 1), y: 11.5, y1: 8, y2: 5 },
  { x: new Date(2000, 0, 1), y: 14, y1: 6, y2: 1.5 },
  { x: new Date(2005, 0, 1), y: 10, y1: 3.5, y2: 2.5 },
  { x: new Date(2010, 0, 1), y: 16, y1: 7, y2: 3.7 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    title: 'Years',
    lineStyle: { width: 0 },
    labelFormat: 'y',
    intervalType: 'Years',
    valueType: 'DateTime',
    edgeLabelPlacement: 'Shift'
  },
  primaryYAxis: {
    title: 'Percentage (%)',
    minimum: 0, maximum: 20, interval: 2,

```

```

        labelFormat: '{value}%'
    },
    series: [
        {
            type: 'StepLine',
            dataSource: chartData, xName: 'x', yName: 'y',
            width: 2, name: 'China',
            marker: {
                visible: true, width: 10, height: 10
            },
        },
        {
            type: 'StepLine',
            dataSource: chartData, xName: 'x', yName: 'y1',
            width: 2, name: 'Australia',
            marker: {
                visible: true, width: 10, height: 10
            },
        },
        {
            type: 'StepLine',
            dataSource: chartData, xName: 'x', yName: 'y2',
            width: 2, name: 'Japan',
            marker: {
                visible: true, width: 10, height: 10
            },
        },
    ],
    //Title for chart
    title: 'Unemployment Rates 1975-2010',
    titleStyle: {
        position: 'Custom',
        x: 300,
        y: 60
    }
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="container">

```

```

        <div id="element"></div>
            <div id="element1"></div>
        </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Title alignment

You can align the title to the near, far, or center of the chart using the [textAlignment](#) property.

INDEX.JS

```

var chartData = [
    { x: new Date(1975, 0, 1), y: 16, y1: 10, y2: 4.5 },
    { x: new Date(1980, 0, 1), y: 12.5, y1: 7.5, y2: 5 },
    { x: new Date(1985, 0, 1), y: 19, y1: 11, y2: 6.5 },
    { x: new Date(1990, 0, 1), y: 14.4, y1: 7, y2: 4.4 },
    { x: new Date(1995, 0, 1), y: 11.5, y1: 8, y2: 5 },
    { x: new Date(2000, 0, 1), y: 14, y1: 6, y2: 1.5 },
    { x: new Date(2005, 0, 1), y: 10, y1: 3.5, y2: 2.5 },
    { x: new Date(2010, 0, 1), y: 16, y1: 7, y2: 3.7 }
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        title: 'Years',
        lineStyle: { width: 0 },
        labelFormat: 'y',
        intervalType: 'Years',
        valueType: 'DateTime',
        edgeLabelPlacement: 'Shift'
    },
    primaryYAxis: {
        title: 'Percentage (%)',
        minimum: 0, maximum: 20, interval: 2,
        labelFormat: '{value}%'
    },
    series: [
        {
            type: 'StepLine',
            dataSource: chartData, xName: 'x', yName: 'y',
            width: 2, name: 'China',
            marker: {
                visible: true, width: 10, height: 10
            }
        },
        {
            type: 'StepLine',
            dataSource: chartData, xName: 'x', yName: 'y1',
            width: 2, name: 'Australia',
            marker: {

```

```

        visible: true, width: 10, height: 10
    },
    },
    {
        type: 'StepLine',
        dataSource: chartData, xName: 'x', yName: 'y2',
        width: 2, name: 'Japan',
        marker: {
            visible: true, width: 10, height: 10
        },
    },
    },
],
//Title for chart
title: 'Unemployment Rates 1975-2010',
titleStyle: { position: 'Bottom', textAlignment: 'Far' },
legendSettings: {visible: false}
}, '#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
    <div id="element1"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Title wrap

The `textStyle` property of chart title provides options to customize the `size`, `color`, `fontFamily`, `fontWeight`, `fontStyle`, `opacity`, `textAlignment` and `textOverflow`.

INDEX.JS

```

var chartData = [
    { month: 'Jan', sales: 35 }, { month: 'Feb', sales: 28 },
    { month: 'Mar', sales: 34 }, { month: 'Apr', sales: 32 },
    { month: 'May', sales: 40 }, { month: 'Jun', sales: 32 },
    { month: 'Jul', sales: 35 }, { month: 'Aug', sales: 55 },
    { month: 'Sep', sales: 38 }, { month: 'Oct', sales: 30 },
    { month: 'Nov', sales: 25 }, { month: 'Dec', sales: 32 }
];
var chart = new ej.charts.Chart({
    // Title for chart
    title: 'Sales Analysis',
    primaryXAxis: {
        valueType: 'Category',
        title: 'Month'
    },
    primaryYAxis: {
        labelFormat: '${value}K'
    },
    series:[{
        dataSource: chartData,
        name: 'Sales',
        xName: 'month',
        yName: 'sales',
        type: 'Line'
    }],
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```


See also

- [Customize the series points using patterns](#)

<!-- markdownlint-disable MD036 -->

Render methods in ##Platform_Name## Chart control

Chart uses following two rendering methods.

- SVG
- Canvas

SVG

SVG is used to render Chart by default for all browsers except IE8 and old versions.

Canvas

You can switch between SVG and Canvas rendering by using the `enableCanvas` option. The canvas mode rendering is used in the following scenarios,

- Plotting large number of data points.
- Performing high frequency live updates.

Limitations

- Animation is not supported.

Accessibility in ##Platform_Name## Chart control

The Chart control followed the accessibility guidelines and standards, including [ADA](#), [Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Chart control is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support | |

| [Section 508](#) Support | |

| Screen Reader Support | |

| Right-To-Left Support | |

| Color Contrast | |

```

| Mobile Device Support |  |

| Keyboard Navigation Support |  |

| Accessibility Checker Validation |  |

| Axe-core Accessibility Validation |  |

<style>

.post .post-content img {
display: inline-block;
margin: 0.5em 0;
}

</style>

<div> - All
features of the control meet the requirement.</div>

<div> - Some features of the control do not meet the requirement.</div>

<div> - The control does not meet the requirement.</div>

```

WAI-ARIA attributes

The Chart control followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Chart control:

- img (role)
- button (role)
- region (role)
- aria-label (attribute)
- aria-hidden (attribute)
- aria-pressed (attribute)

Keyboard interaction

The Chart control followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Chart control.

| **Press** | **To do this** |

| --- | --- |

| **Alt + J** | Moves the focus to the chart element. |

| **Tab** | Moves the focus to the next element in the chart. |

- | Shift + Tab | Moves the focus to the previous element in the chart. |
- | Down Arrow | Moves the focus to the data point left side from the selected point. |
- | Up Arrow | Moves the focus to the data point right side from the selected point. |
- | Left Arrow | Moves the focus to the next series in the chart. |
- | Right Arrow | Moves the focus to the previous series in the chart. |
- | ESC | Cancel the tooltip for the data point. |
- | Enter/Space | Selects the data point in the series. |
- | Down/Left Arrow | Moves the focus to the legend left side from the selected legend. |
- | Up/Right Arrow | Moves the focus to the legend right side from the selected legend. |
- | Enter/Space | Toggles the visibility of the corresponding series. |
- | Ctrl + + | Zoom in the chart. |
- | Ctrl + - | Zoom out the chart. |
- | Down/Up Arrow | Pan the chart vertically. |
- | Left/Right Arrow | Pan the chart horizontally. |
- | R | Reset the zoomed chart. |
- | Ctrl + P | Prints the Chart. |

Ensuring accessibility

The Chart control's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Chart control is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Chart control with accessibility tools.

See also

- [Accessibility in Syncfusion ##Platform_Name## controls](#)

Internationalization in ##Platform_Name## Chart control

Chart provide supports for internationalization for below chart elements.

- Datalabel.
- Axis label.
- Tooltip.

For more information about number and date formatter you can refer [internationalization](#).

<!-- markdownlint-disable MD036 -->

Globalization

Globalization is the process of designing and developing an component that works in different cultures/locales. Internationalization library is used to globalize number, date, time values in Chart component using `labelFormat` property in axis.

Numeric Format

In the below example axis, point and tooltip labels are globalized to EUR.

INDEX.JS

```
var chartData = [
  { x: 1900, y: 4, y1: 2.6 }, { x: 1920, y: 3.0, y1: 2.8 },
  { x: 1940, y: 3.8, y1: 2.6 }, { x: 1960, y: 3.4, y1: 3 },
  { x: 1980, y: 3.2, y1: 3.6 }, { x: 2000, y: 3.9, y1: 3 }
]
var chart = new ej.charts.Chart({
  primaryXAxis: {
    title: 'Year',
    edgeLabelPlacement: 'Shift'
  },
  primaryYAxis: {
    title: 'Sales Amount in Millions',
    //Label format as currency
    labelFormat: 'c'
  },
  series:[{
    dataSource: chartData,
    xName: 'x', yName: 'y',
    name: 'Product X', type: 'Column',
    marker: { dataLabel: { visible: true }}
  },{
    dataSource: chartData,
    xName: 'x', yName: 'y1',
    name: 'Product Y', type: 'Column',
    marker: { dataLabel: { visible: true }}
  }],
  title: 'Average Sales Comparison',
  tooltip: { enable: true, format: '${series.name} <br>${point.x} :
  ${point.y}' }
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
```

```
<body>

  <div id="container">
    <div id="element"></div>
  </div>
</body>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Localization in ##Platform_Name## Chart control

Localization library allows to localize the default text content of Chart. In Chart component, it has the static text on some features(like zooming toolbars) and this can be changed to any other culture(Arabic, Deutsch, French, etc) by defining the locale value and translation object.

<!-- markdownlint-disable MD033 -->

Locale key words	Text to display
Zoom	Zoom
ZoomIn	ZoomIn
ZoomOut	ZoomOut
Reset	Reset
Pan	Pan
ResetZoom	Reset Zoom

To load translation object in an application use load function of L10n class.

For more information about localization, refer this [localization](#).

INDEX.JS

```
var chartData = [
  { x: 1900, y: 4, y1: 2.6 }, { x: 1920, y: 3.0, y1: 2.8 },
  { x: 1940, y: 3.8, y1: 2.6 }, { x: 1960, y: 3.4, y1: 3 },
  { x: 1980, y: 3.2, y1: 3.6 }, { x: 2000, y: 3.9, y1: 3 }
]
ej.base.L10n.load({
  'ar-AR': {
    'chart': {
      ZoomIn: 'تكبير',
      ZoomOut: 'تصغير',
      Zoom: 'زوم',
      Pan: 'مقللة',
      Reset: 'إعادة تعيين',
      ResetZoom: 'زوم إعادة تعيين'
    }
  }
});
```

```

    },
  },
});
var chart = new ej.charts.Chart({
  primaryXAxis: {
    title: 'Year',
    edgeLabelPlacement: 'Shift'
  },
  primaryYAxis: {
    title: 'Sales Amount in Millions',
  },
  series: [{
    dataSource: chartData,
    xName: 'x', yName: 'y',
    name: 'Product X', type: 'Column',
    marker: { dataLabel: { visible: true } }
  }, {
    dataSource: chartData,
    xName: 'x', yName: 'y1',
    name: 'Product Y', type: 'Column',
    marker: { dataLabel: { visible: true } }
  }],
  title: 'Average Sales Comparison',
  locale: 'ar-AR',
  zoomSettings: {
    enableMouseWheelZooming: true,
    enableDeferredZooming: true,
    enablePinchZooming: true,
    enableSelectionZooming: true
  },
}, '#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
```

```

    ele.style.visibility = "visible";
  }
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Ej1 api migration in ##Platform_Name## Chart control

This article describes the API migration process of Chart component from Essential JS 1 to Essential JS 2.

Annotations

<!-- markdownlint-disable MD033 -->

Behaviour	API in Essential JS 1	API in Essential JS 2
rotation of annotation	Property:annotations.angle\$("#chart").ejChart({ annotations: [{angle: 90}]});	Property:annotations.rotationlet chart: Chart = new Chart({annotations: [{rotation: 90}]});chart.appendTo('#chart');
annotations	Property:annotations.content\$("#chart").ejChart({ annotations: [{ content: 'Chart'}]});	Property:annotations.contentlet chart: Chart = new Chart({annotations: [{content: 'Chart'}]});chart.appendTo('#chart');
coordinate unit for annotation	Property:annotations.coordinateUnit\$("#container").ejChart({ annotations :[{ coordinateUnit : "pixels"}]});	Property:annotations.coordinateUnitslet chart: Chart = new Chart({annotations: [{coordinateUnits: 'Pixel'}]});chart.appendTo('#chart');
horizontalAlignment for annotation	Property:annotations.horizontalAlignment\$("#container").ejChart({ annotations :[{ horizontalAlignment : "middle"}]});	Property:annotations.horizontalAlignmentlet chart: Chart = new Chart({annotations: [{horizontalAlignment: 'Center'}]});chart.appendTo('#chart');
margin for annotation	Property:annotations.margin\$("#container").ejChart({ annotations :[{ margin { right: 5, left: 5, top: 5, bottom: 5}}]});	Not applicable
Opacity for annotation	Property:annotations.opacity\$("#container").ejChart({ annotations :[{ opacity: 0.4 }]});	Not applicable

Region for annotation with respect to chart or series	Property:annotations.region \$("#container").ejChart({ annotations : [{ region : "chart" }] });	Property:annotations.region let chart: Chart = new Chart({ annotations: [{ region: 'Chart' }] }); chart.appendTo('#chart');
verticalAlignm ent for annotation	Property:annotations.verticalAlignment \$("#container").ejChart({ annotations : [{ verticalAlignment : "middle" }] });	Property:annotations.verticalAlignm ent let chart: Chart = new Chart({ annotations: [{ verticalAlignment: 'Center' }] }); chart.appendTo('#chart');
Visibility of annotations	Property:annotations.visible \$("#container").ejChart({ annotations : [{ visible: true }] });	Not applicable
X offset for annotation	Property:annotations.x \$("#container").ejChart({ annotations : [{ x : "100" }] });	Property:annotations.x let chart: Chart = new Chart({ annotations: [{ x: '100' }] }); chart.appendTo('#chart');
X axis name in which annotation to be rendered	Property:annotations.xAxisName \$("#container").ejChart({ annotations : [{ xAxisName : "xAxis" }] });	Property:annotations.xAxisName let chart: Chart = new Chart({ annotations: [{ xAxisName: 'xAxis' }] }); chart.appendTo('#chart');
Y offset for annotation	Property:annotations.y \$("#container").ejChart({ annotations : [{ y : "100" }] });	Property:annotations.y let chart: Chart = new Chart({ annotations: [{ y: '100' }] }); chart.appendTo('#chart');
Y axis name in which annotation to be rendered	Property:annotations.yAxisName \$("#container").ejChart({ annotations : [{ yAxisName : "yAxis" }] });	Property:annotations.yAxisName let chart: Chart = new Chart({ annotations: [{ yAxisName: 'yAxis' }] }); chart.appendTo('#chart');

Columns

<!-- markdownlint-disable MD033 -->

Behaviour	API in Essential JS 1	API in Essential JS 2
Columns in chart	Property:columnDefinitions\$("#chart").ejChart({ columnDefinitions: [] });	Property:columns let chart: Chart = new Chart({ columns: [] }); chart.appendTo('#chart');
unit	Property:unit \$("#container").ejChart({	Not Applicable

	<code>columnDefinitions :[{unit : "percentage"}]]});</code>	
width of columns in chart	<code>Property:columnWidth\$("#chart").ejChart({ columnDefinitions: [{ columnWidth: '50%'}]]});</code>	<code>Property:widthlet chart: Chart = new Chart({ columns: [{ width: '300'}]]});chart.appendTo('#chart');</code>
Line customization	<code>Property:lineColor, lineWidth\$("#chart").ejChart({ columnDefinitions: [{ columnWidth: '50%', lineColor: 'brown', lineWidth: 2}]]});</code>	<code>Property:borderlet chart: Chart = new Chart({ columns: [{ width: '300', border: { width: 2, color: 'brown'}}]]});chart.appendTo('#chart');</code>

CommonSeriesOptions

<!-- markdownlint-disable MD033 -->

Behaviour	API in Essential JS 1	API in Essential JS 2
commonSeriesOptions	<code>Property:commonSeriesOptions \$("#container").ejChart({ commonSeriesOptions: { }});</code>	Not Applicable

Crosshair

<!-- markdownlint-disable MD033 -->

Behaviour	API in Essential JS 1	API in Essential JS 2
crosshair	<code>Property:visible\$("#container").ejChart({ crosshair: { visible: true}});</code>	<code>Property:enablelet chart: Chart = new Chart({ crosshair: { enable: false }});chart.appendTo('#chart');</code>
trackballTooltipSettings	<code>Property:trackballTooltipSettings\$("#container").ejChart({ crosshair: { trackballTooltipSettings: { border: { width:2 }}}});</code>	Not applicable

marker	Property:marker\$("#container").ejChart({ crosshair : { marker : { border : { width :2 } } }});	Not applicable
crosshair line style	Property:line\$("#container").ejChart({ crosshair : { line: { width: 2, color: 'red' } }});	let chart: Chart = new Chart({ crosshair: { border: { width: 2, color: 'black' }}});
type	Property:type\$("#container").ejChart({ crosshair : { type: 'trackball' }});	Not applicable
		## 3D chart
		Behaviour API in Essential JS 1 API in Essential JS 2
		3d chart Property:enable3D\$("#container").ejChart({ enable3D: true}); Not applicable
		Rotation of 3d chart Property:enableRotation\$("#container").ejChart({ enableRotation: false}); Not applicable
		depth Property:depth\$("#container").ejChart({ depth: 45}); Not applicable
		## Canvas rendering
		Behaviour API in Essential JS 1 API in Essential JS 2
		canvas rendering Property:enableCanvasRendering\$("#container").ejChart({ enableCanvasRendering: true}); Not applicable
		## Indicators
		Behaviour API in Essential JS 1 API in Essential JS 2

		Type of Indicator	Property:type\$("#container").ejChart({ indicators: [{ type: 'Sma' }] });	Property:typetlet chart: Chart = new Chart({ indicators: [{ type: 'Sma' }] });
		Period for indicator	Property:period\$("#container").ejChart({ indicators: [{ period: 14 }] });	Property:typetlet chart: Chart = new Chart({ indicators: [{ period: 14 }] });
		Period for indicator	Property:period\$("#container").ejChart({ indicators: [{ period: 14 }] });	Property:periodlet chart: Chart = new Chart({ indicators: [{ period: 14 }] });
		%K value in stochastic indicator	Property:kPeriod\$("#container").ejChart({ indicators: [{ kPeriod: 14 }] });	Property:kPeriodlet chart: Chart = new Chart({ indicators: [{ kPeriod: 14 }] });
		%D value in stochastic indicator	Property:dPeriod\$("#container").ejChart({ indicators: [{ dPeriod: 3 }] });	Property:dPeriodlet chart: Chart = new Chart({ indicators: [{ dPeriod: 3 }] });
		Shows overSold/over Bought values	Not applicable	Property:showZoneslet chart: Chart = new Chart({ indicators: [{ showZones: true }] });
		Overbought value for RSI and stochastic indicator	Not applicable	Property:overBoughtlet chart: Chart = new Chart({ indicators: [{ overBought: 80 }] });
		Oversold value for RSI and stochastic indicator	Not applicable	Property:overSoldlet chart: Chart = new Chart({ indicators: [{ overSold: 20 }] });
		Standard deviation for Bollingerbands	Property:standardDeviations\$("#container").ejChart({ indicators: [{ standardDeviations: 2 }] });	Property:standardDeviationlet chart: Chart = new Chart({ indicators: [{

			<code>standardDeviation : 2 }]]]);</code>
	standard deviation	<code>Property:standardDeviations\$("#container").ejChart({ indicators: [{ standardDeviations: 2 }]]]);</code>	<code>Property:standardDeviationlet chart: Chart = new Chart({ indicators: [{ standardDeviation : 2 }]]]);</code>
	Field for indicator	<code>Property:field\$("#container").ejChart({ indicators: [{ field: 'Close' }]]]);</code>	<code>Property:fieldlet chart: Chart = new Chart({ indicators: [{ field: 'Close' }]]]);</code>
	Slow period for MACD indicator	<code>Property:shortPeriod\$("#container").ejChart({ indicators: [{ shortPeriod: 12 }]]]);</code>	<code>Property:slowPeriodlet chart: Chart = new Chart({ indicators: [{ slowPeriod: 12 }]]]);</code>
	Fast period for MACD indicator	<code>Property:longPeriod\$("#container").ejChart({ indicators: [{ longPeriod: 26 }]]]);</code>	<code>Property:fastPeriodlet chart: Chart = new Chart({ indicators: [{ fastPeriod: 26 }]]]);</code>
	Line style for MACD indicator	<code>Property:macdLine\$("#container").ejChart({ indicators: [{ macdLine: { width: 2, fill: 'red' } }]]]);</code>	<code>Property:fastPeriodlet chart: Chart = new Chart({ indicators: [{ macdLine: { width: 2, color: 'red' } }]]]);</code>
	Type of MACD indicator	<code>Property:macdType\$("#container").ejChart({ indicators: [{ macdType: 'both' }]]]);</code>	<code>Property:fastPeriodlet chart: Chart = new Chart({ indicators: [{ macdType: 'Both' }]]]);</code>
	Color of the positive bars in Macd indicators	Not applicable	<code>Property:macdPositiveColorlet chart: Chart = new Chart({ indicators: [{ macdPositiveColor : 'red' }]]]);</code>

		Color of the negative bars in Macd indicators	Not applicable	Property:macdNegativeColor let chart: Chart = new Chart({ indicators: [{ macdNegativeColor: 'red' }] });
		Color for Bollinger bands	Not applicable	Property:bandColor let chart: Chart = new Chart({ indicators: [{ bandColor: 'red' }] });
		Appearance of upper line in indicator	Property:upperLine\$("#container").ejChart({ indicators: [{ upperLine: { fill: '#EECCAA', width: 2 } }] });	Property:upperLine let chart: Chart = new Chart({ indicators: [{ upperLine: { type: 'Smooth', color: '#FFEEFF', width: 2, dashArray: '10,5' } }] });
		Appearance of lower line in indicator	Property:lowerLine\$("#container").ejChart({ indicators: [{ lowerLine: { fill: '#EECCAA', width: 2 } }] });	Property:lowerLine let chart: Chart = new Chart({ indicators: [{ lowerLine: { type: 'Smooth', color: '#FFEEFF', width: 2, dashArray: '10,5' } }] });
		Appearance of period line in indicator	Property:periodLine\$("#container").ejChart({ indicators: [{ periodLine: { fill: '#EECCAA', width: 2 } }] });	Property:lowerLine let chart: Chart = new Chart({ indicators: [{ lowerLine: { type: 'Smooth', color: '#FFEEFF', width: 2, dashArray: '10,5' } }] });
		Name of the series for which indicator has to be drawn.	Property:seriesName\$("#container").ejChart({ indicators: [{ seriesName: '' }] });	Property:lowerLine let chart: Chart = new Chart({ indicators: [{ seriesName: '' }] });

		Options to customize the histogram in MACD indicator	Property:seriesName\$("#container").ejChart({ indicators: [{ histogram: { } }]});	Not applicable
		Enabling animation	Property:enableAnimation\$("#container").ejChart({ indicators: [{ enableAnimation: true }]});	Property:animation.enablelet chart: Chart = new Chart({ indicators: [{ animation: { enable: true } }]});
		Animation duration	Property:animationDuration\$("#container").ejChart({ indicators: [{ animationDuration: 3000 }]});	Property:animation.durationlet chart: Chart = new Chart({ indicators: [{ animation: { duration: 3000 } }]});
		Tooltip	Property:tooltip\$("#container").ejChart({ indicators: [{ tooltip: { visible: true } }]});	Not applicable
		Trigger value of MACD indicator.	Property:trigger\$("#container").ejChart({ indicators: [{ trigger: 14 }]});	Not applicable
		Fill color for indicator	Property:fill\$("#container").ejChart({ indicators: [{ fill: '#EEDDCC' }]});	Property:animation.enablelet chart: Chart = new Chart({ indicators: [{ fill: 'red' }]});
		Width for indicator	Property:width\$("#container").ejChart({ indicators: [{ width: 2 }]});	Property:widthlet chart: Chart = new Chart({ indicators: [{ width: 3 }]});
		xAxis Name of indicator	Property:xAxisName\$("#container").ejChart({ indicators: [{ xAxisName: '' }]});	Property:xAxisNamelet chart: Chart = new Chart({ indicators: [{ xAxisName: '' }]});

		yAxis Name of indicator Property: yAxisName\$("#container").ejChart({ indicators: [{ yAxisName: ' ' }] });	Property: yAxisNamelet chart: Chart = new Chart({ indicators: [{ yAxisName: ' ' }] });
		Visibility of indicator Property: visibility\$("#container").ejChart({ indicators: [{ visibility: true }] });	Not applicable
## Legend			
	Behavior	API in Essential JS 1	API in Essential JS 2
	Default legend	Property: visible\$("#container").ejChart({ legend: { visible: true } });	Property: visiblelet chart: Chart = new Chart({ legendSettings: { visible: true } });
	Legend height	Property: size.height\$("#container").ejChart({ legend: { size: { height: 50 } } });	Property: heightlet chart: Chart = new Chart({ legendSettings: { height: '30' } });
	Legend width	Property: size.width\$("#container").ejChart({ legend: { size: { width: 20 } } });	Property: widthlet chart: Chart = new Chart({ legendSettings: { width: '30' } });
	Legend location in chart	Property: location\$("#container").ejChart({ legend: { location: { x: 3, y: 45 } } });	Property: heightlet chart: Chart = new Chart({ legendSettings: { location: { x: 3, y: 45 } } });
	Legend position in chart	Property: position\$("#container").ejChart({ legend: { position: 'top' } });	Property: positionlet chart: Chart = new Chart({ legendSettings: { position: 'Top' } });
	Legend padding	Not applicable	Property: paddinglet chart: Chart = new Chart({ legendSettings: { padding: 8 } });

		Legend alignment	Property:position\$("#container").ejChart({ legend: { alignment: 'center' } }));	Property:positionlet chart: Chart = new Chart({ legendSettings: { alignment: 'Center' } }));
		text style for legend	Property:font\$("#container").ejChart({ legend: { font: { fontFamily: '', fontWeight: '400', fontStyle: 'italic', size: '12px' } } }));	Property:textStylelet chart: Chart = new Chart({ legendSettings: { textStyle: { size: '12px', color: 'red', fontFamily: 'Italic', fontWeight: '400', fontStyle: 'Normal', opacity: 1, textAlignment: 'Center', textOverflow: 'Trim' } } }));
		shape height of legend	Property:itemStyle.height\$("#container").ejChart({ legend: { itemStyle: { height: 20 } } }));	Property:shapeHeightlet chart: Chart = new Chart({ legendSettings: { shapeHeight: 20 } }));
		shape width of legend	Property:itemStyle.width\$("#container").ejChart({ legend: { itemStyle: { width: 20 } } }));	Property:shapeWidthlet chart: Chart = new Chart({ legendSettings: { shapeWidth: 20 } }));
		shape width of legend	Property:itemStyle.width\$("#container").ejChart({ legend: { itemStyle: { width: 20 } } }));	Property:shapeWidthlet chart: Chart = new Chart({ legendSettings: { shapeWidth: 20 } }));
		shape border of legend	Property:itemStyle.border\$("#container").ejChart({ legend: { itemStyle: { border: { width: 2, color: 'red' } } } }));	Not Applicable
		shape padding of legend	Property:itemPadding\$("#container").ejChart({ legend: { itemPadding: 10 } }));	Property:shapePaddinglet chart: Chart = new Chart({ legendSettings: { shapePadding: 20 } }));

		Background of legend	Property:background\$("#container").ejChart({ legend: { background: 'transparent' }});	Property:backgroundlet chart: Chart = new Chart({ legendSettings: { background: 'transparent' }});
		Opacity of legend	Property:opacity\$("#container").ejChart({ legend: { opacity: 0.3 }});	Property:opacitylet chart: Chart = new Chart({ legendSettings: { opacity: 0.4 }});
		Toggle visibility of series while legend click	Property:toggleSeriesVisibility\$("#container").ejChart({ legend: { toggleSeriesVisibility: true }});	Property:toggleVisibilitylet chart: Chart = new Chart({ legendSettings: { toggleVisibility: true }});
		Title for legend	Property:title\$("#container").ejChart({ legend: { title: { text: 'LegendTitle', font: { }, textAlign: 'middle' } }});	Not applicable
		Text Overflow for legend	Property:title\$("#container").ejChart({ legend: { textOverFlow: 'trim' }});	Property:textStyle.textOverflowlet chart: new Chart({ legend: { text: { textOverFlow: 'trim' } }});
		Text width for legend while setting text overflow	Property:textWidth\$("#container").ejChart({ legend: { textWidth: 20 }});	Not applicable
		Scroll bar for legend	Property:enableScrollBar\$("#container").ejChart({ legend: { enableScrollBar: true }});	Not applicable
		Row count for legend	Property:rowCount\$("#container").ejChart({ legend: { rowCount: 2 }});	Not applicable

		Column count for legend	Property:columnCount\$("#container").ejChart({ legend: { columnCount: 2 }});	Not applicable
		Color for legend items	Property:fill\$("#container").ejChart({ legend: { fill: '#EEFFCC' }});	Not applicable
		## primaryXAxis		
		Behavior	API in Essential JS 1	API in Essential JS 2
		Alternate grid band	Property:alternateGridBand\$("#container").ejChart({ primaryXAxis: { alternateGridBand: { even: { fill: 'red' }}}});	Not applicable
		Axis line crosses value	Property:crossesAt\$("#container").ejChart({ primaryXAxis: { crossesAt: 0 }});	Property:crossesAtlet chart: Chart = new Chart({ primaryXAxis: { crossesAt: 4}});chart.appendTo('#chart');
		axis name with which the axis line has to be	Property:crossesInAxis\$("#container").ejChart({ primaryXAxis: { crossesInAxis: '' }});	Property:crossesInAxislet chart: Chart = new Chart({ primaryXAxis: { crossesInAxis: '' }});chart.appendTo('#chart');

		crossed		
		axis elements placed with axis line	Property:showNextToAxisLine <code>\$("#container").ejChart({ primaryXAxis: { showNextToAxisLine : true } });</code>	Property:placeNextToAxisLine <code>let chart: Chart = new Chart({ primaryXAxis: { placeNextToAxisLine: '' } });chart.appendTo('#chart');</code>
		axis line style	Property:axisLine.color <code>\$("#container").ejChart({ primaryXAxis: { axisLine: { color : 'red' } } });</code>	Property:lineStyle.color <code>let chart: Chart = new Chart({ primaryXAxis: { lineStyle: { color: 'black' } } });chart.appendTo('#chart');</code>
		axis line dash array	Property:axisLine.color <code>\$("#container").ejChart({ primaryXAxis: { axisLine: { dashArray : '10, 5' } } });</code>	Property:lineStyle.dashArray <code>let chart: Chart = new Chart({ primaryXAxis: { lineStyle: { dashArray: '10, 5' } } });chart.appendTo('#chart');</code>
		Offset for axis	Property:axisLine.offset <code>\$("#container").ejChart({ primaryXAxis: { axisLine: { offset : 10 } } });</code>	Property:plotOffset <code>let chart: Chart = new Chart({ primaryXAxis: { plotOffset: 10 } });chart.appendTo('#chart');</code>
		Visible of an axis	Property:axisLine.offset <code>\$("#container").ejChart({ primaryXAxis: { axisLine: { visible :</code>	Property:visible <code>let chart: Chart = new Chart({ primaryXAxis: { visible: false } });chart.appendTo('#chart');</code>

			<code>false } }});</code>	
		Width of an axis	<code>Property:axisLine.width\$("#container").ejChart({ primaryXAxis: { axisLine: { width : 2 } } });</code>	<code>Property:lineStyle.widthlet chart: Chart = new Chart({ primaryXAxis: { lineStyle: { width: 3 } } });chart.appendTo('#chart');</code>
		Column index of an axis	<code>Property:columnIndex\$("#container").ejChart({ primaryXAxis: { columnIndex : 2 } });</code>	<code>Property:columnIndexlet chart: Chart = new Chart({ primaryXAxis: { columnIndex: 2 } });chart.appendTo('#chart');</code>
		span of an axis to place horizontally or vertically	<code>Property:columnSpan\$("#container").ejChart({ primaryXAxis: { columnIndex : 2 } });</code>	<code>Property:spanlet chart: Chart = new Chart({ primaryXAxis: { span: 2 } });chart.appendTo('#chart');</code>
		Crosshair label of an axis	<code>Property:crossHairLabel.visible\$("#container").ejChart({ primaryXAxis: { crossHairLabel: { visible: true } } });</code>	<code>Property:crossHairTooltip.enablelet chart: Chart = new Chart({ primaryXAxis: { crossHairTooltip: { enable: true } } });chart.appendTo('#chart');</code>
		Crosshair	Not applicable	<code>Property:crossHairTooltip.filllet chart: Chart = new Chart({ primaryXAxis: { crossHairTooltip: { fill: 'red' } } });chart.appendTo('#chart');</code>

		label color of an axis		
		Crosshair label text style	Not applicable	Property:crossHairTooltip.textStylelet chart: Chart = new Chart({ primaryXAxis: { crossHairTooltip: { textStyle: { } } } });chart.appendTo('#chart');
		Desired interval count for primaryX axis	Property:desiredIntervals\$("#container").ejChart({ primaryXAxis: { desiredIntervals: 4 } });	Property:desiredIntervalslet chart: Chart = new Chart({ primaryXAxis: { desiredIntervals: 4 } });chart.appendTo('#chart');
		Edge label placement for primaryX axis	Property:edgeLabelPlacement\$("#container").ejChart({ primaryXAxis: { edgeLabelPlacement: 'none' } });	Property:edgeLabelPlacementlet chart: Chart = new Chart({ primaryXAxis: { edgeLabelPlacement: 'Shift' } });chart.appendTo('#chart');
		Enable trim for axis labels	Property:enableTrim\$("#container").ejChart({ primaryXAxis: { enableTrim: true } });	Property:enableTrimlet chart: Chart = new Chart({ primaryXAxis: { enableTrim: true } });chart.appendTo('#chart');
		Specify	Property:enableAutoIntervalOnZooming\$("#container")	Property:enableAutoIntervalOnZoominglet chart: Chart = new Chart({ primaryXAxis: {

		the interval of the axis according to the zoomed data of the chart	<pre>.ejChart({ primaryXAxis: { enableAutoIntervalOnZooming: true }});</pre>	<pre>enableAutoIntervalOnZooming: true });chart.appendTo('#chart');</pre>
		Specifies the interval of the axis according to the zoomed data of the chart	<pre>Property:enableAutoIntervalOnZooming\$("#container") .ejChart({ primaryXAxis: { enableAutoIntervalOnZooming: true }});</pre>	<pre>Property:enableAutoIntervalOnZoominglet chart: Chart = new Chart({ primaryXAxis: { enableAutoIntervalOnZooming: true }});chart.appendTo('#chart');</pre>
		Font style for	<pre>Property:font\$("#container").ejChart({ primaryXAxis: { font:</pre>	<pre>Property:titleStylelet chart: Chart = new Chart({ primaryXAxis: { titleStyle: { } }});chart.appendTo('#chart');</pre>

		primaryXAxis	{ fontFamily: 'Calibri', fontStyle: 'italic', fontWeight: '', opacity: 0.5, size: 12} }));	
		Indexed for category axis	Property:isIndexed\$("#container").ejChart({ primaryXAxis: { isIndexed: true }));	Property:isIndexedlet chart: Chart = new Chart({ primaryXAxis: { isIndexed: true });chart.appendTo('#chart');
		Interval type for date time axis	Property:intervalType\$("#container").ejChart({ primaryXAxis: { intervalType: 'Auto' }});	Property:intervalTypelet chart: Chart = new Chart({ primaryXAxis: { intervalType: 'Auto });chart.appendTo('#chart');
		Inversed axis	Property:isInversed\$("#container").ejChart({ primaryXAxis: { isInversed: true }));	Property:isInversedlet chart: Chart = new Chart({ primaryXAxis: { isInversed: true });chart.appendTo('#chart');
		Custom label format	Property:labelFormat\$("#container").ejChart({ primaryXAxis: { labelFormat: '{value}K' }});	Property:labelFormatlet chart: Chart = new Chart({ primaryXAxis: { labelFormat: '{value}K' });chart.appendTo('#chart');

		labelIntersection	Property:labelIntersection \$("#container").ejChart({ primaryXAxis: { labelIntersection: 'trim' }}});	Property:labelIntersection let chart: Chart = new Chart({ primaryXAxis: { labelIntersection: 'Trim' }});chart.appendTo('#chart');
		labelPosition	Property:labelPosition \$("#container").ejChart({ primaryXAxis: { labelPosition: 'inside' }}});	Property:labelPosition let chart: Chart = new Chart({ primaryXAxis: { labelPosition: 'Inside' }});chart.appendTo('#chart');
		labelPlacement for category axis	Property:labelPlacement \$("#container").ejChart({ primaryXAxis: { labelPlacement: 'onTicks' }}});	Property:labelPlacement let chart: Chart = new Chart({ primaryXAxis: { labelPlacement: 'OnTicks' }});chart.appendTo('#chart');
		Axis label alignment	Property:alignment \$("#container").ejChart({ primaryXAxis: { alignment: 'center' }}});	Not Applicable
		Rotation of axis labels	Property:labelRotation \$("#container").ejChart({ primaryXAxis: { labelRotation: 45 }}});	Property:labelRotation let chart: Chart = new Chart({ primaryXAxis: { labelRotation: 45 }});chart.appendTo('#chart');
		Log base	Property:logBase \$("#container").ejChart({ primaryXAxis: { logBase: 10 }});	Property:labelRotation let chart: Chart = new Chart({ primaryXAxis: { logBase: 10 }});chart.appendTo('#chart');

		value for logarithmic axis	<pre>rt({ primaryXAxis: { logBase: 10 }});</pre>	
		Major grid line	<pre>Property:majorGridLines.visible\$("#container").ejChart({ primaryXAxis: { majorGridLines: { visible: true } }});</pre>	Not Applicable
		Width of Major Grid Lines	<pre>Property:majorGridLines.width\$("#container").ejChart({ primaryXAxis: { majorGridLines: { width: 2 } }});</pre>	<pre>Property:majorGridLines.widthlet chart: Chart = new Chart({ primaryXAxis: { majorGridLines: { width: 2 } }});chart.appendTo('#chart');</pre>
		Color of Major Grid Lines	<pre>Property:majorGridLines.color\$("#container").ejChart({ primaryXAxis: { majorGridLines: { color: 'black' } }});</pre>	<pre>Property:majorGridLines.colorlet chart: Chart = new Chart({ primaryXAxis: { majorGridLines: { color: 'black' } }});chart.appendTo('#chart');</pre>
		Dash Array of Major Grid Lines	<pre>Property:majorGridLines.dashArray\$("#container").ejChart({ primaryXAxis: { majorGridLines: { dashArray: 'black' } }});</pre>	<pre>Property:majorGridLines.dashArraylet chart: Chart = new Chart({ primaryXAxis: { majorGridLines: { dashArray: 'black' } }});chart.appendTo('#chart');</pre>

			<code>dashArray: 'black' } }});</code>	
	Opacity of major grid line	Property:majorGridLines.opacity\$("#container").ejChart({ primaryXAxis: { majorGridLines: { opacity: true} } }));	Not Applicable	
	Major Tick line	Property:majorTickLines.visible\$("#container").ejChart({ primaryXAxis: { majorTickLines: { visible: true} } }));	Not Applicable	
	Width of Major Tick Lines	Property:majorTickLines.width\$("#container").ejChart({ primaryXAxis: { majorTickLines: { width: 2} } }));	Property:majorTickLines.widthlet chart: Chart = new Chart({ primaryXAxis: { majorTickLines: { width: 2} } });chart.appendTo('#chart');	
	Height of Major Tick Lines	Property:majorTickLines.size\$("#container").ejChart({ primaryXAxis: { majorTickLines: { size: 2} } }));	Property:majorTickLines.heightlet chart: Chart = new Chart({ primaryXAxis: { majorTickLines: { height: 2} } });chart.appendTo('#chart');	
	Color of	Property:majorTickLines.color\$("#contain	Property:majorTickLines.colorlet chart: Chart = new Chart({ primaryXAxis: { majorTickLines: { color: 'black' } } });chart.appendTo('#chart');	

		Major Tick Lines	er").ejChart({ primaryXAxis: { majorTickLines: { color: 'black' } }});	
		Opacity of major Tick line	Property:majorTickLines.opacity\$("#container").ejChart({ primaryXAxis: { majorTickLines: { opacity: true} } }});	Not Applicable
		maximum labels of primary XAxis	Property:maximumLabels\$("#container").ejChart({ primaryXAxis: { maximumLabels: 5 } }});	Property:maximumLabelslet chart: Chart = new Chart({ primaryXAxis: { maximumLabels: 4 } });chart.appendTo('#chart');
		maximum label width of primary XAxis to trim	Property:maximumLabelWidth\$("#container").ejChart({ primaryXAxis: { maximumLabelWidth: 40 } }});	Property:maximumLabelWidthlet chart: Chart = new Chart({ primaryXAxis: { maximumLabelWidth: 4 } });chart.appendTo('#chart');
		minor grid line	Property:minorGridLines.visible\$("#container").ejChart({	Not Applicable

			<pre>primaryXAxis: { minorGridLines: { visible: true} }));</pre>	
		Width of minorGridLines	<pre>Property:minorGridLines.width \$("#container").ejChart({ primaryXAxis: { minorGridLines: { width: 2} } });</pre>	<pre>Property:minorGridLines.widthlet chart: Chart = new Chart({ primaryXAxis: { minorGridLines: { width: 2} } });chart.appendTo('#chart');</pre>
		Color of minorGridLines	<pre>Property:minorGridLines.color \$("#container").ejChart({ primaryXAxis: { minorGridLines: { color: 'black' } } });</pre>	<pre>Property:minorGridLines.colorlet chart: Chart = new Chart({ primaryXAxis: { minorGridLines: { color: 'black' } } });chart.appendTo('#chart');</pre>
		DashArray of minorGridLines	<pre>Property:minorGridLines.dash Array\$("#container").ej Chart({ primaryXAxis: { minorGridLines: { dashArray: 'black' } } });</pre>	<pre>Property:minorGridLines.dashArraylet chart: Chart = new Chart({ primaryXAxis: { minorGridLines: { dashArray: 'black' } } });chart.appendTo('#chart');</pre>
		Opacity of minor grid line	<pre>Property:minorGridLines.opaci ty\$("#contai ner").ejCha rt({ primaryXAxis: { minorGridLi nes: {</pre>	Not Applicable

			opacity: true} }));	
	min or Tick line	Property:minor TickLines.visibl e\$("#contai ner").ejCha rt({ primaryXAxis: { minorTickLi nes: { visible: true} }));	Not Applicable	
	Wid th of min orTi ckLi nes	Property:minor TickLines.width \$("#contain er").ejChar t({ primaryXAxis: { minorTickLi nes: { width: 2} }});	Property:minorTickLines.widthlet chart: Chart = new Chart({ primaryXAxis: { minorTickLines: { width: 2} } });chart.appendTo('#chart');	
	Hei ght of min orTi ckLi nes	Property:minor TickLines.size\$ \$("#containe r").ejChart (({ primaryXAxis: { minorTickLi nes: { size: 2} }});	Property:minorTickLines.heightlet chart: Chart = new Chart({ primaryXAxis: { minorTickLines: { height: 2} } });chart.appendTo('#chart');	
	Col or of min orTi ckLi nes	Property:minor TickLines.color \$("#contain er").ejChar t({ primaryXAxis: { minorTickLi nes: { color: 'black' } }});	Property:minorTickLines.colorlet chart: Chart = new Chart({ primaryXAxis: { minorTickLines: { color: 'black' } } });chart.appendTo('#chart');	
	Opa city of	Property:minor TickLines.opaci ty\$("#contai	Not Applicable	

		minor Tick line	ner").ejChart({ primaryXAxis: { minorTickLines: { opacity: true} }}};	
		Minor ticks per interval of primaryXAxis	Property:minorTicksPerInterval let chart: Chart = new Chart({ primaryXAxis: { minorTicksPerInterval: 4 }});chart.appendTo('#chart');	
		name of the primaryXAxis	Property:name let chart: Chart = new Chart({ primaryXAxis: { name: 'primaryXAxis' }});chart.appendTo('#chart');	
		Orientation of primaryXAxis	Property:orientation let chart: Chart = new Chart({ primaryXAxis: { orientation: 'Vertical' }});	Not Applicable
		Plot offset for primaryXAxis	Property:plotOffset let chart: Chart = new Chart({ primaryXAxis: { plotOffset: 0 }});chart.appendTo('#chart');	

		minimum for primaryXAxis	Property:range.minimum\$("#container").ejChart({ primaryXAxis: { range: { minimum: 10 }}});	Property:minimumlet chart: Chart = new Chart({ primaryXAxis: { minimum: 23 }});chart.appendTo('#chart');
		maximum for primaryXAxis	Property:range.maximum\$("#container").ejChart({ primaryXAxis: { range: { maximum: 10 }}});	Property:maximumlet chart: Chart = new Chart({ primaryXAxis: { maximum: 23 }});chart.appendTo('#chart');
		interval for primaryXAxis	Property:range.interval\$("#container").ejChart({ primaryXAxis: { range: { interval: 1 }}});	Property:intervallet chart: Chart = new Chart({ primaryXAxis: { interval: 2 }});chart.appendTo('#chart');
		RangePadding for primaryXAxis	Property:rangePadding\$("#container").ejChart({ primaryXAxis: { rangePadding: 'None' }}});	Property:rangePaddinglet chart: Chart = new Chart({ primaryXAxis: { rangePadding: 'None' }});chart.appendTo('#chart');
		Rounding Places in primaryXAxis	Property:roundingPlaces\$("#container").ejChart({ primaryXAxis: { roundingPlaces: 3 }});	Property:labelFormatlet chart: Chart = new Chart({ primaryXAxis: { labelFormat: 'n3' }});chart.appendTo('#chart');
		Scrollbar	Property:scrollbarSettings	Not Applicable

		ar sett ings of pri mar yXA xis	<code>\$("#container").ejChart({ primaryXAxis: { scrollbarSettings : { } } });</code>	
		Tick Posi tion in pri mar yXA xis	<code>Property:tickLinesPosition\$(" #container").ejChart({ primaryXAxis: { tickLinesPosition: 'Inside' } });</code>	<code>Property:tickPositionlet chart: Chart = new Chart({ primaryXAxis: { tickPosition: 'Inside' } });chart.appendTo('#chart');</code>
		valu eTy pe of pri mar yXA xis	<code>Property:valueType\$(" #container").ejChart({ primaryXAxis: { valueType: 'DateTime' } });</code>	<code>Property:valueTypelet chart: Chart = new Chart({ primaryXAxis: { valueType: 'DateTime' } });chart.appendTo('#chart');</code>
		visi ble of pri mar yXA xis	<code>Property:visible\$(" #container").ejChart({ primaryXAxis: { visible: true } });</code>	<code>Property:visiblelet chart: Chart = new Chart({ primaryXAxis: { visible: true } });chart.appendTo('#chart');</code>
		zoo mFa ctor of pri mar yXA xis	<code>Property:zoomFactor\$(" #container").ejChart({ primaryXAxis: { zoomFactor: 0.3 } });</code>	<code>Property:zoomFactorlet chart: Chart = new Chart({ primaryXAxis: { zoomFactor: 0.3 } });chart.appendTo('#chart');</code>
		zoo mP ositi on	<code>Property:zoomPosition\$(" #container").ejChart({ primaryXAxis:</code>	<code>Property:zoomPositionlet chart: Chart = new Chart({ primaryXAxis: { zoomPosition: 0.3 } });chart.appendTo('#chart');</code>

		of pri mar yXA xis	s: { zoomPositio n: 0.3 }});	
		labe lBor der of pri mar yXA xis	Property:labelB order\$("#con tainer").ej Chart({ primaryXAxi s: { labelBorder : { color: 'red', width: 2} }});	Property:borderlet chart: Chart = new Chart({ primaryXAxis: { border: { color: 'red', width: 3 } }});chart.appendTo('#chart');
		title of pri mar yXA xis	Property:title.t ext\$("#conta iner").ejCh art({ primaryXAxi s: { title: { text: 'Chart title' } }});	Property:titlelet chart: Chart = new Chart({ primaryXAxis: { title: 'Chart title' }});chart.appendTo('#chart');
		Stri plin e of pri mar yXA xis	Property:stripLi ne\$("#conta iner").ejCh art({ primaryXAxi s: { stripLine: [] }});	Property:stripLineslet chart: Chart = new Chart({ primaryXAxis: { stripLines: [] }});chart.appendTo('#chart');
		Mul tilev el labe ls of pri mar yXA xis	Property:multiL evelLabels\$("# container").ejChart({ primaryXAxi s: { multiLevelL abels: [] }});	Property:multiLevelLabelslet chart: Chart = new Chart({ primaryXAxis: { stripLines: [] }});chart.appendTo('#chart');
		skel eto n for an	Not Applicable	Property:skeletonlet chart: Chart = new Chart({ axes: [{ skeleton: 'yMd' }}];chart.appendTo('#chart');

		axes		
		skel eto n typ e for an axe s	Not Applicable	<p>Property:skeletonType <pre>let chart: Chart = new Chart({ axes: [{ skeletonType: 'DateTime' }]});chart.appendTo('#chart');</pre></p> <p>## primaryYAxis</p> <p>Behavior API in Essential JS 1 API in Essential JS 2</p> <p>Alternate grid band Property:alternateGridBand <pre>\$("#container").ejChart ({ primaryYAxis: { alternateGridBand: { even: { fill: 'red }}}});</pre> Not applicable</p> <p>Axis line cross value Property:crossesAt <pre>\$("#container").ejChart({ primaryYAxis: { crossesAt: 0 });</pre> Property:crossesAt <pre>let chart: Chart = new Chart({ primaryYAxis: { crossesAt: 4});chart.app endTo('#chart');</pre></p> <p>axis name with which the axis line has to be crossed Property:crossesInAxis <pre>\$("#container").ejChart({ primaryYAxis: { crossesInAxis: '' });</pre> Property:crossesInAxis <pre>let chart: Chart = new Chart({ primaryYAxis: { crossesInAxis: '' });chart.appe ndTo('#chart') ;</pre></p> <p>axis element s placed with axis line Property:showNextToAxisLine <pre>\$("#container").ejChart ({ primaryYAxis: { showNextToAxisLine : true }});</pre> Property:placeNextToAxisLine <pre>let chart: Chart = new Chart({ primaryYAxis: { placeNextToAxisLine: '' });chart.appe ndTo('#chart') ;</pre></p> <p>axis line style Property:axisLine.color <pre>\$("#container").ejChart({</pre> Property:lineStyle.color <pre>let chart:</pre></p>

				<pre> primaryYAxis: { axisLine: { color : 'red' } })); </pre>	<pre> Chart = new Chart({ primaryYAxis: { lineStyle: { color: 'black' } });chart.appe ndTo('#chart') ; </pre>
			axis line dashArr ay	<pre> Property:axisLine.color\$("#co ntainer").ejChart({ primaryYAxis: { axisLine: { dashArray : '10, 5' } })); </pre>	<pre> Property:lineStyle. dashArraylet chart: Chart = new Chart({ primaryYAxis: { lineStyle: { dashArray: '10, 5' } });chart.appe ndTo('#chart') ; </pre>
			Offset for axis	<pre> Property:axisLine.offset\$("##c ontainer").ejChart({ primaryYAxis: { axisLine: { offset : 10 } })); </pre>	<pre> Property:plotOffse tlet chart: Chart = new Chart({ primaryYAxis: { plotOffset: 10 });chart.appe ndTo('#chart') ; </pre>
			Visible of an axis	<pre> Property:axisLine.offset\$("##c ontainer").ejChart({ primaryYAxis: { axisLine: { visible : false } })); </pre>	<pre> Property:visiblele t chart: Chart = new Chart({ primaryYAxis: { visible: false });chart.appe ndTo('#chart') ; </pre>
			Width of an axis	<pre> Property:axisLine.width\$("##c ontainer").ejChart({ primaryYAxis: { axisLine: { width : 2 } }); </pre>	<pre> Property:lineStyle. widthlet chart: Chart = new Chart({ primaryYAxis: { lineStyle: { width: 3 } });chart.appe ndTo('#chart') ; </pre>

				<p>Column index of an axis</p> <p>Property:columnIndex\$("#container").ejChart({ primaryYAxis: { columnIndex: 2 }});</p> <p>span of an axis to place horizontally or vertically</p> <p>Property:columnSpan\$("#container").ejChart({ primaryYAxis: { columnIndex: 2 }});</p> <p>Crosshair label of an axis</p> <p>Property:crossHairLabel.visible\$("#container").ejChart({ primaryYAxis: { crossHairLabel: { visible: true }}});</p> <p>Crosshair label color of an axis</p> <p>Not applicable</p> <p>Crosshair label text style</p> <p>Not applicable</p>	<p>Property:columnIndex</p> <pre>let chart: Chart = new Chart({ primaryYAxis: { columnIndex: 2 });chart.append To('#chart') ;</pre> <p>Property:span</p> <pre>let chart: Chart = new Chart({ primaryYAxis: { span: 2 });chart.append To('#chart') ;</pre> <p>Property:crossHair Tooltip.enable</p> <pre>let chart: Chart = new Chart({ primaryYAxis: { crossHairToolt ip: { enable: true }}});chart.app endTo('#chart');</pre> <p>Property:crossHair Tooltip.fill</p> <pre>let chart: Chart = new Chart({ primaryYAxis: { crossHairToolt ip: { fill: 'red' }}});chart.app endTo('#chart');</pre> <p>Property:crossHair Tooltip.textStyle</p> <pre>1 et chart: Chart = new Chart({ primaryYAxis: { crossHairToolt ip: {</pre>
--	--	--	--	--	--

				<pre> textStyle: { } }}});chart.appendTo('#chart')); Property:desiredIn tervalslet chart: Chart = new Chart({ primaryYAxis: { desiredInterva ls: 4 }});chart.appe ndTo('#chart') ; Property:edgeLab elPlacementlet chart: Chart = new Chart({ primaryYAxis: { edgeLabelPlace ment: 'Shift' }});chart.appe ndTo('#chart') ; Property:enableTri mlet chart: Chart = new Chart({ primaryYAxis: { enableTrim: true }});chart.appe ndTo('#chart') ; Property:enableAu toIntervalOnZoom inglet chart: Chart = new Chart({ primaryYAxis: { enableAutoInte rvalOnZooming: true }});chart.appe ndTo('#chart') ; </pre>
			<p>Desired interval count for primary YAxis</p>	<pre> Property:desiredIntervals\$("## container").ejChart({ primaryYAxis: { desiredIntervals: 4}}); </pre>
			<p>Edges primary YAxis</p>	<pre> Property:edgeLabelPlacement\$ ("#container").ejChart({ primaryYAxis: { edgeLabelPlacement: 'none' }}); </pre>
			<p>Enables trim for axis labels</p>	<pre> Property:enableTrim\$("##cont ainer").ejChart({ primaryYAxis: { enableTrim: true }}); </pre>
			<p>Specifie s the interval of the axis accordin g to the zoomed data of the chart</p>	<pre> Property:enableAutoIntervalO nZooming\$("##container"). ejChart({ primaryYAxis: { enableAutoIntervalOnZoo ming: true }}); </pre>

				<p>Specifies the interval of the axis according to the zoomed data of the chart</p> <p>Property:enableAutoIntervalOnZooming <pre>let chart: Chart = new Chart({ primaryYAxis: { enableAutoIntervalOnZooming: true }});chart.appendTo('#chart') ;</pre></p> <p>Property:enableAutoIntervalOnZooming <pre>Property:enableAutoIntervalOnZooming\$("#container").ejChart({ ejChart({ primaryYAxis: { enableAutoIntervalOnZooming: true }}});</pre></p>	
				<p>Font style for primary YAxis</p> <p>Property:fontStyle <pre>Property:font\$("#container").ejChart({ primaryYAxis: { font: { fontFamily: 'Calibri', fontStyle: 'italic', fontWeight: '', opacity: 0.5, size: 12} }});</pre></p> <p>Property:titleStyle <pre>let chart: Chart = new Chart({ primaryYAxis: { titleStyle: { } }});chart.appendTo('#chart') ;</pre></p>	
				<p>Indexed for category axis</p> <p>Property:isIndexed <pre>let chart: Chart = new Chart({ primaryYAxis: { isIndexed: true }});chart.appendTo('#chart') ;</pre></p> <p>Property:isIndexed <pre>Property:isIndexed\$("#container").ejChart({ primaryYAxis: { isIndexed: true }}});</pre></p>	
				<p>Interval type for date time axis</p> <p>Property:intervalType <pre>let chart: Chart = new Chart({ primaryYAxis: { intervalType: 'Auto }});chart.appendTo('#chart') ;</pre></p> <p>Property:intervalType <pre>Property:intervalType\$("#container").ejChart({ primaryYAxis: { intervalType: 'Auto' }});</pre></p>	
				<p>Inversed axis</p> <p>Property:isInversed <pre>dlet chart: Chart = new Chart({ primaryYAxis: { isInversed: true }});</pre></p> <p>Property:isInversed <pre>Property:isInversed\$("#container").ejChart({ primaryYAxis: { isInversed: true }}});</pre></p>	

				<pre> });chart.appendTo('#chart') ; Property:labelFormat\$("#container").ejChart({ primaryYAxis: { labelFormat: '{value}K' }}});chart.appendTo('#chart') ; Property:labelIntersectAction\$ ("#container").ejChart({ primaryYAxis: { labelIntersectAction: 'trim' }}}); Property:labelPosition\$("#container").ejChart({ primaryYAxis: { labelPosition: 'inside' }}}); Property:labelPlacement\$("#container").ejChart({ primaryYAxis: { labelPlacement: 'onTicks' }}}); Property:alignment\$("#container").ejChart({ primaryYAxis: { </pre>
			Custom label format	<pre> });chart.appendTo('#chart') ; Property:labelFormat\$("#container").ejChart({ primaryYAxis: { labelFormat: '{value}K' }}});chart.appendTo('#chart') ; Property:labelIntersectAction\$ ("#container").ejChart({ primaryYAxis: { labelIntersectAction: 'trim' }}}); Property:labelPosition\$("#container").ejChart({ primaryYAxis: { labelPosition: 'inside' }}}); Property:labelPlacement\$("#container").ejChart({ primaryYAxis: { labelPlacement: 'onTicks' }}}); Property:alignment\$("#container").ejChart({ primaryYAxis: { </pre>
			labelIntersectAction	<pre> });chart.appendTo('#chart') ; Property:labelFormat\$("#container").ejChart({ primaryYAxis: { labelFormat: '{value}K' }}});chart.appendTo('#chart') ; Property:labelIntersectAction\$ ("#container").ejChart({ primaryYAxis: { labelIntersectAction: 'trim' }}}); Property:labelPosition\$("#container").ejChart({ primaryYAxis: { labelPosition: 'inside' }}}); Property:labelPlacement\$("#container").ejChart({ primaryYAxis: { labelPlacement: 'onTicks' }}}); Property:alignment\$("#container").ejChart({ primaryYAxis: { </pre>
			labelPosition	<pre> });chart.appendTo('#chart') ; Property:labelFormat\$("#container").ejChart({ primaryYAxis: { labelFormat: '{value}K' }}});chart.appendTo('#chart') ; Property:labelIntersectAction\$ ("#container").ejChart({ primaryYAxis: { labelIntersectAction: 'trim' }}}); Property:labelPosition\$("#container").ejChart({ primaryYAxis: { labelPosition: 'inside' }}}); Property:labelPlacement\$("#container").ejChart({ primaryYAxis: { labelPlacement: 'onTicks' }}}); Property:alignment\$("#container").ejChart({ primaryYAxis: { </pre>
			labelPlacement for category axis	<pre> });chart.appendTo('#chart') ; Property:labelFormat\$("#container").ejChart({ primaryYAxis: { labelFormat: '{value}K' }}});chart.appendTo('#chart') ; Property:labelIntersectAction\$ ("#container").ejChart({ primaryYAxis: { labelIntersectAction: 'trim' }}}); Property:labelPosition\$("#container").ejChart({ primaryYAxis: { labelPosition: 'inside' }}}); Property:labelPlacement\$("#container").ejChart({ primaryYAxis: { labelPlacement: 'onTicks' }}}); Property:alignment\$("#container").ejChart({ primaryYAxis: { </pre>
			Axis label	<pre> });chart.appendTo('#chart') ; Property:labelFormat\$("#container").ejChart({ primaryYAxis: { labelFormat: '{value}K' }}});chart.appendTo('#chart') ; Property:labelIntersectAction\$ ("#container").ejChart({ primaryYAxis: { labelIntersectAction: 'trim' }}}); Property:labelPosition\$("#container").ejChart({ primaryYAxis: { labelPosition: 'inside' }}}); Property:labelPlacement\$("#container").ejChart({ primaryYAxis: { labelPlacement: 'onTicks' }}}); Property:alignment\$("#container").ejChart({ primaryYAxis: { </pre>

			<p>alignment alignment: 'center' } } };</p> <p>Rotation of axis labels Property:labelRotation\$("#container").ejChart({ primaryYAxis: { labelRotation: 45 } } });</p> <p>Log base value for logarithmic axis Property:logBase\$("#container").ejChart({ primaryYAxis: { logBase: 10 } } });</p> <p>Major grid line Property:majorGridLines.visible\$("#container").ejChart({ primaryYAxis: { majorGridLines: { visible: true } } } });</p> <p>Width of MajorGridLines Property:majorGridLines.width\$("#container").ejChart({ primaryYAxis: { majorGridLines: { width: 2 } } } });</p> <p>Color of MajorGridLines Property:majorGridLines.color\$("#container").ejChart({ primaryYAxis: { majorGridLines: { color: 'black' } } } });</p>	<p>Property:labelRotation let chart: Chart = new Chart({ primaryYAxis: { labelRotation: 45 } });chart.appendTo('#chart');</p> <p>Property:labelRotation let chart: Chart = new Chart({ primaryYAxis: { logBase: 10 } });chart.appendTo('#chart');</p> <p>Not Applicable</p> <p>Property:majorGridLines.width let chart: Chart = new Chart({ primaryYAxis: { majorGridLines: { width: 2 } } });chart.appendTo('#chart');</p> <p>Property:majorGridLines.color let chart: Chart = new Chart({ primaryYAxis: { majorGridLines: { color: 'black' } } });chart.appendTo('#chart');</p>
--	--	--	--	--

				<pre>ndTo('#chart') ;</pre>	
				<pre>Property:majorGridLines.dashArray1 let chart: Chart = new Chart({ primaryYAxis: { majorGridLines : { dashArray: 'black' } });chart.append ndTo('#chart') ;</pre>	
			DashArray of MajorGridLines	<pre>Property:majorGridLines.dashArray\$("#container").ejChart({ primaryYAxis: { majorGridLines: { dashArray: 'black' } } });</pre>	
			Opacity of major grid line	<pre>Property:majorGridLines.opacity\$("#container").ejChart({ primaryYAxis: { majorGridLines: { opacity: true } } });</pre>	Not Applicable
			Major Tick line	<pre>Property:majorTickLines.visible\$("#container").ejChart({ primaryYAxis: { majorTickLines: { visible: true } } });</pre>	Not Applicable
			Width of MajorTickLines	<pre>Property:majorTickLines.width\$("#container").ejChart({ primaryYAxis: { majorTickLines: { width: 2 } } });</pre>	<pre>Property:majorTickLines.widthlet chart: Chart = new Chart({ primaryYAxis: { majorTickLines : { width: 2 } });chart.append ndTo('#chart') ;</pre>
			Height of MajorTickLines	<pre>Property:majorTickLines.size\$("#container").ejChart({ primaryYAxis: { majorTickLines: { size: 2 } } });</pre>	<pre>Property:majorTickLines.heightlet chart: Chart = new Chart({ primaryYAxis: { majorTickLines : { height: 2 } });chart.append ndTo('#chart') ;</pre>

				<div>Property:majorTickLines.colorlet chart: Chart = new Chart({ primaryYAxis: { majorTickLines : { color: 'black' } });chart.appendTo('#chart') ;</div>
			<div>Property:majorTickLines.opacity\$("#container").ejChart({ primaryYAxis: { majorTickLines: { opacity: true} } }));</div>	Not Applicable
			<div>Property:maximumLabels\$("#container").ejChart({ primaryYAxis: { maximumLabels: 5 } }));</div>	<div>Property:maximumLabelslet chart: Chart = new Chart({ primaryYAxis: { maximumLabels: 4 });chart.appendTo('#chart') ;</div>
			<div>Property:maximumLabelWidth\$("#container").ejChart({ primaryYAxis: { maximumLabelWidth: 40 } }));</div>	<div>Property:maximumLabelWidthlet chart: Chart = new Chart({ primaryYAxis: { maximumLabelWidth: 4 });chart.appendTo('#chart') ;</div>
			<div>Property:minorGridLines.visible\$("#container").ejChart({ primaryYAxis: { minorGridLines: { visible: true} } }));</div>	Not Applicable
			<div>Property:minorGridLines.width\$("#container").ejChart({ primaryYAxis: { minorGridLines: { width: 2} } }));</div>	<div>Property:minorGridLines.widthlet chart: Chart = new Chart({ primaryYAxis:</div>

				<pre> { minorGridLines : { width: 2} });chart.appendTo('#chart') ; Property:minorGridLines.color let chart: Chart = new Chart({ primaryYAxis: { minorGridLines : { color: 'black' } });chart.appendTo('#chart') ; Property:minorGridLines.dashArray let chart: Chart = new Chart({ primaryYAxis: { minorGridLines : { dashArray: 'black' } });chart.appendTo('#chart') ; Property:minorGridLines.opacity y\$("#container").ejChart({ primaryYAxis: { minorGridLines: { opacity: true} })); Property:minorTickLines.visible \$("#container").ejChart({ primaryYAxis: { minorTickLines: { visible: true} })); Property:minorTickLines.width let chart: Chart = new Chart({ primaryYAxis: { minorTickLines : { width: 2} });chart.appendTo('#chart') ; </pre>
			<p>Color of minorGridLines</p>	<pre> Property:minorGridLines.color \$("#container").ejChart({ primaryYAxis: { minorGridLines: { color: 'black' } } }); </pre>
			<p>DashArray of minorGridLines</p>	<pre> Property:minorGridLines.dashArray \$("#container").ejChart({ primaryYAxis: { minorGridLines: { dashArray: 'black' } } }); </pre>
			<p>Opacity of minor grid line</p>	<p>Not Applicable</p>
			<p>minor Tick line</p>	<p>Not Applicable</p>
			<p>Width of minorTickLines</p>	<pre> Property:minorTickLines.width \$("#container").ejChart({ primaryYAxis: { minorTickLines: { width: 2 } } }); </pre>

				<pre> ndTo('#chart') ; Property:minorTickLines.heightlet chart: Chart = new Chart({ primaryYAxis: { minorTickLines: { size: minorTickLines : { height: 2} }});chart.append ndTo('#chart') ; </pre>
Height of minor TickLines	Property:minorTickLines.size\$			<pre> ("#container").ejChart({ primaryYAxis: { minorTickLines: { size: minorTickLines 2} })); </pre>
Color of minor TickLines	Property:minorTickLines.color\$			<pre> ("#container").ejChart({ primaryYAxis: { minorTickLines: { color: 'black' } } }); </pre>
Opacity of minor Tick line	Property:minorTickLines.opacity\$	Not Applicable		<pre> ("#container").ejChart({ primaryYAxis: { minorTickLines: { opacity: true} } }); </pre>
Minor ticks per interval of primary YAxis	Property:minorTicksPerInterval\$			<pre> ("#container").ejChart(({ primaryYAxis: { minorTicksPerInterval: 4 } } }); </pre>
name of the primary YAxis	Property:name\$			<pre> ("#container").ejChart({ primaryYAxis: { name: 'primaryYAxis' } }); </pre>

				<pre>ndTo('#chart') ;</pre>
			Orientat ion of primary YAxis	<pre>Property:orientation\$("#cont ainer").ejChart({ primaryYAxis: { orientation: 'Vertical' }});</pre>
			Plot offset for primary YAxis	<pre>Property:plotOffset\$("#cont ainer").ejChart({ primaryYAxis: { plotOffset: 0 }});</pre>
			minimu m for primary YAxis	<pre>Property:range.minimum\$("# container").ejChart({ primaryYAxis: { range: { minimum: 10 }}});</pre>
			maximu m for primary YAxis	<pre>Property:range.maximum\$("# container").ejChart({ primaryYAxis: { range: { maximum: 10 }}});</pre>
			interval for primary YAxis	<pre>Property:range.interval\$("#co ntainer").ejChart({ primaryYAxis: { range: { interval: 1 }}});</pre>
			RangeP adding for	<pre>Property:rangePadding\$("#co ntainer").ejChart({ primaryYAxis: { rangePadding: 'None' }});</pre>

Not Applicable

Property:plotOffse

```
tlet chart:  
Chart = new  
Chart({  
primaryYAxis:  
{ plotOffset:  
0  
}});chart.appe  
ndTo('#chart')  
;
```

Property:minimu

```
mlet chart:  
Chart = new  
Chart({  
primaryYAxis:  
{ minimum: 23  
}});chart.appe  
ndTo('#chart')  
;
```

Property:maximu

```
mlet chart:  
Chart = new  
Chart({  
primaryYAxis:  
{ maximum: 23  
}});chart.appe  
ndTo('#chart')  
;
```

Property:interval1

```
et chart:  
Chart = new  
Chart({  
primaryYAxis:  
{ interval: 2  
}});chart.appe  
ndTo('#chart')  
;
```

Property:rangePad

```
dinglet chart:  
Chart = new  
Chart({  
primaryYAxis:
```

			<p>primary YAxis</p> <pre>{ rangePadding: 'None' });chart.appendTo('#chart') ; Property:labelFormat let chart: Chart = new Chart({ primaryYAxis: { labelFormat: 'n3' });chart.appendTo('#chart') ; Property:roundingPlaces \$("#container").ejChart(({ primaryYAxis: { roundingPlaces: 3 }}); Scrollbar Property:scrollbarSettings \$("#container").ejChart(({ primaryYAxis: { scrollbarSettings : { } }}); Not Applicable Property:tickPosition let chart: Chart = new Chart({ primaryYAxis: { tickPosition: 'Inside' });chart.appendTo('#chart') ; Property:valueType let chart: Chart = new Chart({ primaryYAxis: { valueType: 'DateTime' });chart.appendTo('#chart') ; Property:visible let chart: Chart = new Chart({ primaryYAxis: { visible: true } });</pre>
			<p>Roundin g Places in primary YAxis</p>
			<p>ScrollBa r settings of primary YAxis</p>
			<p>TickPosi tion in primary YAxis</p>
			<p>valueTy pe of primary YAxis</p>
			<p>visible of primary YAxis</p>

				<pre> });chart.appendTo('#chart') ; Property:zoomFactor let chart: Chart = new Chart({ primaryYAxis: { zoomFactor: 0.3 });chart.appendTo('#chart') ; Property:zoomPosition let chart: Chart = new Chart({ primaryYAxis: { zoomPosition: 0.3 });chart.appendTo('#chart') ; Property:border let chart: Chart = new Chart({ primaryYAxis: { border: { color: 'red', width: 3 } });chart.appendTo('#chart') ; Property:title let chart: Chart = new Chart({ primaryYAxis: { title: { title: { text: 'Chart title' } 'Chart title' });chart.appendTo('#chart') ; Property:stripLine let chart: Chart = new Chart({ primaryYAxis: { stripLines: [] </pre>
			<pre> zoomFactor of primary YAxis Property:zoomFactor\$("#container").ejChart({ primaryYAxis: { zoomFactor: 0.3 }); </pre>	<pre> zoomPosition of primary YAxis Property:zoomPosition\$("#container").ejChart({ primaryYAxis: { zoomPosition: 0.3 }); </pre>
			<pre> labelBorder of primary YAxis Property:labelBorder\$("#container").ejChart({ primaryYAxis: { labelBorder: { color: 'red', width: 2 } }); </pre>	
			<pre> title of primary YAxis Property:title.text\$("#container").ejChart({ primaryYAxis: { title: { text: 'Chart title' } }); </pre>	
			<pre> StripLine of primary YAxis Property:stripLine\$("#container").ejChart({ primaryYAxis: { stripLine: [] }); </pre>	

				<pre> });chart.appendTo('#chart') ; Property:multiLevelLabelslet chart: Chart = new Chart({ primaryYAxis: { stripLines: [] });chart.appendTo('#chart') ; Property:skeleton let chart: Chart = new Chart({ axes: [{ skeleton: 'yMd' }]);chart.appendTo('#chart')); Property:skeleton Typelet chart: Chart = new Chart({ axes: [{ skeletonType: 'DateTime' }]);chart.appendTo('#chart')); ## Axes Behavior API in Essential JS 1 API in Essential JS 2 Property:alternateGridBand AlternateGridBand \$("#container").ejChart({ axes: [{ alternateGridBand: { even: { fill: 'red' }}}]); </pre>
--	--	--	--	---

			<p>Axis Property:crossesAt</p> <pre> line esAt\$("#container").ejChart({ crossesAt: [{ 4}]]});chart.appendTo('#chart' valu crossesAt: 0 }]]]); </pre> <p>axis name with this Property:crossesInAxis</p> <pre> whi esInAxis\$("#container").ejChart({ ch ontainer").ejChart({ the axes: [{ crossesInAxis: '' axis crossesInAxis: '' line is: '' has }]]]); to be crossed axis ele Property:showNextToAxisLine me NextToAxisLine nts \$("#container").ejChart({ plac er").ejChart({ ed t({ axes: [wit showNextToAxisLine : h xisLine : axis true }]]]); line </pre> <p>Property:axisLineStyle</p> <pre> axis ne.color\$("#container").ejChart({ line ontainer").ejChart({ styl axes: [{ e axisLine: { color : 'red' } }]]]); </pre> <p>Property:axisLineStyle.dashArray</p> <pre> axis ne.color\$("#container").ejChart({ line ontainer").ejChart({ das axes: [{ axisLine: { dashArray: '10, 5' } }]]]); </pre>
--	--	--	--

			<pre> dashArray : }]);chart.appendTo('#chart') '10, 5' } ; }]); Property:axisLine ne.offset\$("#c ontainer"). ejChart({ axes: [{ axisLine: { offset : 10 } }]); Property:axisLine ne.offset\$("#c ontainer"). ejChart({ axes: [{ axisLine: { visible : false } } }]); Property:axisLine ne.width\$("#c ontainer"). ejChart({ axes: [{ axisLine: { width : 2 } } }]); Property:column nIndex\$("#co ntainer").e jChart({ axes: [{ columnIndex : 2 } }]); Property:column nSpan\$("#con tainer").ej Chart({ axes: [{ columnIndex : 2 } }]); </pre>
--	--	--	--

				<p>vertical y</p> <p>Property:crossHairLabel.visible</p> <pre> e\$("#container").ejChart({ axes: [{ crossHairLabel: { visible: true } }] }); </pre>
				<p>Crosshair label color of an axis</p> <p>Property:crossHairTooltip.fill</p> <pre> let chart: Chart = new Chart({ axes: [{ crossHairTooltip: { fill: 'red' } }] });chart.appendTo('#chart'); </pre> <p>Not applicable</p>
				<p>Crosshair label text style</p> <p>Property:crossHairTooltip.textStyle</p> <pre> let chart: Chart = new Chart({ axes: [{ crossHairTooltip: { textStyle: { } } }] });chart.appendTo('#chart'); </pre> <p>Not applicable</p>
				<p>Desired intervals</p> <p>Property:desiredIntervals</p> <pre> let chart: Chart = new Chart({ axes: [{ desiredIntervals: 4 }] });chart.appendTo('#chart'); </pre>
				<p>Edge labels</p> <p>Property:edgeLabelPlacement</p> <pre> let chart: Chart = new Chart({ axes: [{ edgeLabelPlacement: 'Shift' </pre>

			<pre> mar t({ axes: []});chart.appendTo('#chart') yYA { xis edgeLabelPl acement: 'none' }}}); Ena Property:enabl bles eTrim\$("#con Property:enableTrimlet chart: trim tainer").ej Chart = new Chart({ axes: [{ for Chart({ enableTrim: true axis axes: [{ }]);chart.appendTo('#chart') labe enableTrim: ; ls true }]); Spe cifie s the inte rval of the axis eAutoIntervalO Property:enableAutoIntervalOnZoomi acc nZooming\$("# nglet chart: Chart = new ordi container") Chart({ axes: [{ ng .ejChart({ enableAutoIntervalOnZooming: to axes: [{ true the enableAutoI zoo ntervalOnZo me oming: true d }]); dat a of the cha rt Spe cifie Property:enabl s eAutoIntervalO Property:enableAutoIntervalOnZoomi the nZooming\$("# nglet chart: Chart = new inte container") Chart({ axes: [{ rval .ejChart({ enableAutoIntervalOnZooming: of axes: [{ true the enableAutoI axis ntervalOnZo acc oming: true ordi }]); </pre>
--	--	--	--

			<p>ng to the zoo me d dat a of the cha rt</p> <p>Property:font\$ ("#containe r").ejChart { axes: [{ font: { fontFamily: 'Calibri', fontStyle: 'italic', fontWeight: '', opacity: 0.5, size: 12} }]);</p> <p>Ind exe d for cate gor Y axis</p> <p>Property:isInde xed\$("#conta iner").ejCh art({ axes: [{ isIndexed: true }]});</p> <p>Inte rval typ e for dat e tim e axis</p> <p>Property:interv alType\$("#co ntainer").e jChart({ axes: [{ intervalTyp e}: 'Auto' }]});</p> <p>Inve rse</p> <p>Property:isInve rsed\$("#cont ainer").ejC</p> <p>Property:titleStylelet chart: Chart = new Chart({ axes: [{ titleStyle: { } }]);chart.appendTo('#chart') ;</p> <p>Property:isIndexedlet chart: Chart = new Chart({ axes: [{ isIndexed: true }]);chart.appendTo('#chart') ;</p> <p>Property:intervalTypelet chart: Chart = new Chart({ axes: [{ intervalType: 'Auto' }]);chart.appendTo('#chart') ;</p> <p>Property:isInversedlet chart: Chart = new Chart({ axes: [{ isInversed: true</p>
--	--	--	---

				<pre>d hart({ })});chart.appendTo('#chart') axis axes: [{ ; isInversed: true }]);</pre> <p>Property:labelF</p> <pre>Cus ormat\$("#con tom tainer").ej labe Chart({ Property:labelFormatlet chart: axes: [{ Chart = new Chart({ axes: [{ labelFormat: '{value}K' labelFormat }]);chart.appendTo('#chart') for : ; mat '{value}K' }]);</pre> <p>Property:labelI</p> <pre>labe intersectAction llnt \$("#contain Property:labelIntersectActionlet er").ejChar chart: Chart = new Chart({ t({ axes: [axes: [{ ctA { labelIntersectAction: 'Trim' ctio labelInters }]);chart.appendTo('#chart') n ectAction: ; 'trim' }]);</pre> <p>Property:labelP</p> <pre>osition\$("#co labe ntainer").e Property:labelPositionlet chart: IPos jChart({ Chart = new Chart({ axes: [{ axes: [{ labelPosition: 'Inside' labelPositi }]);chart.appendTo('#chart') itio on: ; n 'inside' }]);</pre> <p>Property:labelP</p> <pre>IPla Placement\$("## cem container") Property:labelPlacementlet chart: ent .ejChart({ Chart = new Chart({ axes: [{ for axes: [{ labelPlacement: 'OnTicks' cate labelPlacem }]);chart.appendTo('#chart') gor ent: ; y 'onTicks' axis }]);</pre> <p>Axis</p> <pre>labe Property:align l ment\$("#con l tainer").ej Not Applicable alig Chart({ nm axes: [{ ent alignment:</pre>
--	--	--	--	--

			<pre> 'center' }}}); Property:labelR otation\$("#co ntainer").e jChart({ axes: [{ labelRotati on: 45 }] }); Log bas e valu e for loga rith mic axis Property:logBa se\$("#contai ner").ejCha rt({ axes: [{ logBase: 10 }] }); Property:labelRotationlet chart: Chart = new Chart({ axes: [{ labelRotation: 45 }] });chart.appendTo('#chart') ; Property:labelRotationlet chart: Chart = new Chart({ axes: [{ logBase: 10 }] });chart.appendTo('#chart') ; Property:major GridLines.visibl e\$("#contai ner").ejCha rt({ axes: [{ majorGridLi nes: { visible: true} }] }); Property:major GridLines.width \$("#contain er").ejChar t({ axes: [{ majorGridLi nes: { width: 2 } }] }); Property:majorGridLines.widthlet chart: Chart = new Chart({ axes: [{ majorGridLines: { width: 2 } }] });chart.appendTo('#chart') ; Property:major GridLines.color \$("#contain er").ejChar t({ axes: [{ majorGridLi nes: { </pre>
--	--	--	---

				<pre> color: 'black' } }}}); Property:major Das GridLines.dash hAr Array\$("#con Property:majorGridLines.dashArrayle ray tainer").ej t chart: Chart = new Chart({ of Chart({ axes: [{ majorGridLines: { Maj axes: [{ dashArray: 'black' } orG majorGridLi }}});chart.appendTo('#chart') ridL nes: { ; dashArray: ines 'black' } }}}); Property:major Opa GridLines.opaci city ty\$("#contai of ner").ejCha maj rt({ axes: Not Applicable or [{ grid majorGridLi line nes: { opacity: true} }]]}); Property:major TickLines.visibl Maj e\$("#contai or ner").ejCha Tick rt({ axes: Not Applicable line [{ majorTickLi nes: { visible: true} }]]}); Property:major Wid TickLines.width th \$("#contain Property:majorTickLines.widthlet of er").ejChar chart: Chart = new Chart({ Maj t({ axes: [axes: [{ majorTickLines: { orTi { width: 2} ckLi majorTickLi }}});chart.appendTo('#chart') nes: { ; nes width: 2} }}}); Property:major Hei TickLines.size\$ Property:majorTickLines.heightlet ght ("#containe chart: Chart = new Chart({ of r").ejChart axes: [{ majorTickLines: { Maj r").ejChart height: 2} orTi ({ axes: [</pre>
--	--	--	--	---

			<pre> ckLi { nes majorTickLi ; nes: { size: 2} }}}); Property:major Col TickLines.color or \$("#contain of er").ejChar Maj t({ axes: [orTi majorTickLi ckLi nes: { nes color: 'black' } }}}); Property:major Opa TickLines.opaci city ty\$("#contai of ner").ejCha maj rt({ axes: or [{ Tick majorTickLi line nes: { opacity: true} }]); max imu Property:maxi m mumLabels\$("# labe #container" ls of).ejChart({ pri axes: [{ mar maximumLabe yYA ls: 5 }]); xis max imu Property:maxi m mumLabelWidt labe h\$("#contai ls ner").ejCha wid rt({ axes: th [{ of maximumLabe pri lWidth: 40 mar }]); yYA xis </pre>
--	--	--	---

			<p>to trim</p> <p>Property:minor GridLines.visibl</p> <p>min or grid line</p> <pre>e\$("#contai ner").ejCha rt({ axes: [{ minorGridLi nes: { visible: true} }]]);</pre> <p>Property:minor GridLines.width</p> <p>Wid th of min orG ridL ines</p> <pre>\$("#contain er").ejChar t({ axes: [{ minorGridLi nes: { width: 2} }]]);</pre> <p>Property:minor GridLines.color</p> <p>Col or of min orG ridL ines</p> <pre>\$("#contain er").ejChar t({ axes: [{ minorGridLi nes: { color: 'black' } }]]);</pre> <p>Property:minor GridLines.dash</p> <p>Das hAr ray of min orG ridL ines</p> <pre>\$("#con tainer").ej Chart({ axes: [{ minorGridLi nes: { dashArray: 'black' } }]]);</pre> <p>Property:minor GridLines.opaci</p> <p>Opa city of min</p> <pre>\$("#contai ner").ejCha rt({ axes:</pre> <p>Not Applicable</p>
--	--	--	---

			<pre> or [{ grid minorGridLi line nes: { opacity: true} }]]); Property:minor TickLines.visibl e\$("#contai ner").ejCha rt({ axes: [{ minorTickLi nes: { visible: true} }]]); Property:minor TickLines.width \$("#contain er").ejChar t({ axes: [{ minorTickLi nes: { width: 2} }]]); Property:minor TickLines.size\$ ("#containe r").ejChart ({ axes: [{ minorTickLi nes: { size: 2} }]]); Property:minor TickLines.color \$("#contain er").ejChar t({ axes: [{ minorTickLi nes: { color: 'black' } }]]); Property:minor TickLines.opaci ty\$("#contai </pre>	<p>Not Applicable</p> <p>Property:minorTickLines.widthlet chart: Chart = new Chart({ axes: [{ minorTickLines: { width: 2} }]]);chart.appendTo('#chart') ;</p> <p>Property:minorTickLines.heightlet chart: Chart = new Chart({ axes: [{ minorTickLines: { height: 2} }]]);chart.appendTo('#chart') ;</p> <p>Property:minorTickLines.colorlet chart: Chart = new Chart({ axes: [{ minorTickLines: { color: 'black' } }]]);chart.appendTo('#chart') ;</p> <p>Not Applicable</p>
--	--	--	---	---

			<pre> min ner").ejCha or rt({ axes: [{ Tick minorTickLi line nes: { opacity: true} }]]}); Min or tick Property:minor s TicksPerInterva per l\$("#contain Property:minorTickLines.colorlet inte er").ejChar chart: Chart = new Chart({ rval t({ axes: [axes: [{ of { minorTicksPerInterval: 4 pri minorTicksP }]]});chart.appendTo('#chart') mar erInterval: ; yYA 4 }]]}); xis na me Property:name of \$("#contain Property:namelet chart: Chart the er").ejChar = new Chart({ axes: [{ name: pri t({ axes: ['primaryYAxis' mar { name: }]]});chart.appendTo('#chart') yYA 'primaryYAx ; xis is' }]]}); Orie Property:orient ntat ation\$("#con ion tainer").ej of Chart({ pri axes: [{ Not Applicable mar orientation yYA 'Vertical' xis }]]}); Plot offs Property:plotO et ffsset\$("#cont Property:plotOffsetlet chart: for ainer").ejC Chart = new Chart({ axes: [{ pri hart({ plotOffset: 0 mar axes: [{ }]]});chart.appendTo('#chart') yYA plotOffset: ; xis 0 }]]}); </pre>
--	--	--	--

			<pre> min Property:range. imu minimum\$("# m container") for .ejChart({ pri axes: [{ mar range: { yYA minimum: 10 xis }]]}); max Property:range. imu maximum\$("# m container") for .ejChart({ pri axes: [{ mar range: { yYA maximum: 10 xis }]]}); inte Property:range. rval interval\$("#co for ntainer").e pri jChart({ mar axes: [{ yYA range: { xis interval: 1 }]]}); Ran geP Property:range add Padding\$("#c ing ontainer"). for ejChart({ pri axes: [{ mar rangePaddin yYA g: 'None' xis }]]}); Rou ndi Property:round ng ingPlaces Plac \$("#contain es er").ejChar in t({ axes: [pri { mar roundingPla yYA ces: 3 xis }]]}); </pre>
			<pre> Property:minimumlet chart: Chart = new Chart({ axes: [{ minimum: 23 }]]);chart.appendTo('#chart') ; Property:maximumlet chart: Chart = new Chart({ axes: [{ maximum: 23 }]]);chart.appendTo('#chart') ; Property:intervallet chart: Chart = new Chart({ axes: [{ interval: 2 }]]);chart.appendTo('#chart') ; Property:rangePaddinglet chart: Chart = new Chart({ axes: [{ rangePadding: 'None' }]]);chart.appendTo('#chart') ; Property:labelFormatlet chart: Chart = new Chart({ axes: [{ labelFormat: 'n3' }]]);chart.appendTo('#chart') ; </pre>

			<p> Scrollbar Property:scrollbarSettings setter\$("#container").ejChart({ axes: [Not Applicable { scrollbarSettings : { } }] }); xis </p> <p> Tick Property:tickLinesPosition\$("#container").ejChart({ axes: [{ tickPosition: 'Inside' }] });chart.appendTo('#chart') xis </p> <p> ValueType Property:valueType\$("#container").ejChart({ axes: [{ valueType: 'DateTime' }] });chart.appendTo('#chart') xis </p> <p> visible Property:visible\$("#container").ejChart({ axes: [{ visible: true }] });chart.appendTo('#chart') xis </p> <p> zoomFactor Property:zoomFactor\$("#container").ejChart({ axes: [{ zoomFactor: 0.3 }] });chart.appendTo('#chart') xis </p> <p> zoomPosition Property:zoomPosition\$("#container").ejChart({ axes: [{ zoomPosition: 0.3 }] });chart.appendTo('#chart') xis </p>
--	--	--	--

			<pre> ositi ejChart({ });chart.appendTo('#chart') on axes: [{ ; of zoomPositio n: 0.3 pri }]); mar yYA xis </pre>
			<pre> labe Property:labelB lBor order\$("#con der tainer").ej of Chart({ Property:borderlet chart: Chart pri axes: [{ = new Chart({ axes: [{ mar labelBorder width: 3 } yYA : { color: });chart.appendTo('#chart') xis 'red', ; width: 2} }); </pre>
			<pre> Property:title.t title ext\$("#conta of iner").ejCh pri art({ axes: mar [{ title: yYA { text: xis 'Chart title' } }); </pre>
			<pre> Stri Property:stripli plin ne\$("#conta e of iner").ejCh pri art({ axes: mar [{ yYA stripLine: xis []]}); </pre>
			<pre> Mul Property:multiL tilev evellabels\$(" el #container" labe).ejChart({ ls of axes: [{ axe multiLevelL s abels: [] }); </pre>
			<pre> skel eto n Not Applicable for an </pre>
			<pre> Property:skeletonlet chart: Chart = new Chart({ axes: [{ skeleton: 'yMd' });chart.appendTo('#chart') ; </pre>

				<p>axes</p> <p>Property:skeletonType</p> <pre>let chart: Chart = new Chart({ axes: [{ skeletonType: 'DateTime' }]});chart.appendTo('#chart') ;</pre> <p>## Rows</p> <p>Behavior</p> <p>API in Essential JS 1</p> <p>API in Essential JS 2</p> <p>Property:rows1</p> <pre>let chart: Chart = new Chart({ rows: []});chart .appendTo('#chart');</pre> <p>Property:rowDefinitions</p> <pre>\$("#chart").ejC Chart({ rowDefiniti ons: []});</pre> <p>Property:unit</p> <pre>\$("#contain er").ejChar t({ rowDefiniti ons :[{unit : "percentage "}]});</pre> <p>Not Applicable</p> <p>unit</p> <p>Not Applicable</p> <p>Property:height</p> <pre>let chart: Chart = new Chart({ rows: [{ height: '300'}];}); chart.appen dTo('#chart ');</pre> <p>Property:rowHeight</p> <pre>\$("#cha rt").ejChar t({ rowDefiniti ons: [{ rowHeight: '50%'}];});</pre> <p>Property:lineColor, lineWidth</p> <pre>\$("#chart").ejC hart({ rowDefiniti ons: [{ rowHeight: '50%',</pre> <p>Property:border</p> <pre>let chart: Chart = new Chart({ rows: [{ height: '300', border: { width: 2,</pre>
--	--	--	--	--

				<pre> lineColor: color: 'brown', 'brown'}}}]; lineWidth: });chart.ap 2}}]); pendTo('#ch art'); ## Series Behaviour API in Essential JS API in 1 Essential JS 2 Property:bea arFillColor1 et chart: Chart = new Chart({ series: [[bearFil lColor: 'red' }}]);});cha rt.append To('#char t'); Property:ro wslet chart: Chart = new Chart({ series: [[border: { color: 'red', width: 2, dashArray: '10, 5' } }}]);});cha rt.append To('#char t'); Property:ro wslet chart: Chart = new Chart({ series: [[boxPlotMo de: 'inclusive' }}]);}); </pre>
--	--	--	--	--

				<pre> 'Inclusive' }]);});chart.append To('#chart'); Property:minimumRadius let chart: Chart = new Chart({ series: [{ minRadius : 2 }]);});chart.append To('#chart'); Property:maximumRadius let chart: Chart = new Chart({ series: [{ maxRadius : 2 }]);});chart.append To('#chart'); Property:bullFillColor let chart: Chart = new Chart({ series: [{bullFillColor: 'red' }]);});chart.append To('#chart'); Card Property:cardinal inal SplineTension\$ ("rdinalSpline </pre>
--	--	--	--	--

				<pre> splin #chart").ejCh e art({ series: tensi [{ on cardinalSpli for eTension: 0.5 splin }]);}); e Chart = serie new s Chart({ series: [{ cardinalS plineTens ion: 0.5 }]);});cha rt.append To('#char t'); Property:co lumnWidth let chart: Chart = new Chart({ series: [{ columnWid th: 0.5 }]);});cha rt.append To('#char t'); Property:co lumnSpacin g let chart: Chart = new Chart({ series: [{ columnSpa cing: 0.5 }]);});cha rt.append To('#char t'); Property:cornerR adius.topLeft\$ (" #chart").ejCh art({ series: [{ topLeft: 0 }]);}); for rect Chart({ series: [{ topLeft: 0 }]);}); </pre>
--	--	--	--	--

				<pre> angl e serie s series: [{ topLeft: 0 }];});cha rt.append To('#char t'); Property:co rnerRadius. topRightle t chart: Chart = new Chart({ series: [{ topRight: 0 }];});cha rt.append To('#char t'); Property:co rnerRadius. bottomRigh tlet chart: Chart = new Chart({ series: [{ bottomRig ht: 0 }];});cha rt.append To('#char t'); Property:co rnerRadius. bottomLeft let chart: Chart = new Chart({ series: [{ bottomLef t: 0 </pre>
--	--	--	--	--

				<pre> serie s Property:dashArraylet chart: Chart = new ejChart({ series: [{ dashArray: '10, 5' }];}); Property:dashArraylet chart: Chart = new ejChart({ series: [{ dashArray: '10, 5' }];});cha rt.append To('#char t'); Data Source for series Property:dataSource ce\$("#chart"). ejChart({ series: [{ dataSource: [] }];}); Property:drawTypelet chart: Chart = new ejChart({ series: [{ drawType: 'Line' }];}); Property:emptyPointSettings.visible Not Applicable \$("#chart").e jChart({ </pre>
--	--	--	--	---

				<pre> s for series: [{ serie emptyPointSet s tings: { visible: false } }];}); Property:emptyPointSettings.displayMode let chart: Chart = new Chart({ series: [{} displayMode: de: 'Average' }];});chart.append To('#chart'); Property:emptyPointSettings.fill let chart: Chart = new Chart({ series: [{} fill: 'red' }];});chart.append To('#chart'); Property:fill let chart: Chart = new Chart({ series: [{} emptyPointSettings : { color: 'red', width: 2 }];}); </pre>
--	--	--	--	---

					<pre> });});chart.append To('#chart'); Property:animation.enablelet chart: Chart = new Chart({ series: [animation : { enable: false }]);});chart.append To('#chart'); Property:animation.durationlet chart: Chart = new Chart({ series: [animation : { duration: 1000 }]);});chart.append To('#chart'); Property:animation.durationlet chart: Chart = new Chart({ series: [animation : { delay: 100 }]);});chart.append </pre>
	Enable animation for series	Property:enableAnimation\$("#chart").ejChart({ series: [{ enableAnimation: true }]});			<pre> });});chart.append To('#chart'); Property:enableAnimation\$("#chart").ejChart({ series: [{ enableAnimation: true }]}); Property:animationDuration\$("#chart").ejChart ({ series: [{ animationDuration: 1000 }]}); Property:animationDuration\$("#chart").ejChart ({ series: [{ animationDuration: 1000 }]}); Property:animationDuration\$("#chart").ejChart ({ series: [{ animationDuration: 1000 }]}); </pre>
	Animation delay for series	Not Applicable			<pre> });});chart.append To('#chart'); Property:animationDuration\$("#chart").ejChart ({ series: [{ animationDuration: 1000 }]}); Property:animationDuration\$("#chart").ejChart ({ series: [{ animationDuration: 1000 }]}); Property:animationDuration\$("#chart").ejChart ({ series: [{ animationDuration: 1000 }]}); </pre>

				<pre> To('#chart'); Drag Property:dragSettings\$("#chart").ejChart({ series: [{ dragSettings: { mode: 'X' } }]);}); Error Property:errorBarSettings\$("#chart").ejChart({ series: [{ errorBarSettings: { series: [{errorBarSettings: { } } } } }]);}); Closed Property:isClosed\$("#chart").ejChart({ series: [{ isClosed: true }]);}); Stacking Property:isStacking\$("#chart").ejChart({ series: [{ isStacking: true }]);}); Line Property:lineCap\$("#chart").ejChart({ series: [{ lineCap: 'butt' }]);}); Line Property:lineCap\$("#chart").ejChart({ series: [{ lineCap: 'butt' }]);}); </pre>
--	--	--	--	--

				<pre> series series: [{ s lineJoin: 'round' }];}); Property:er rorBarSetti ngslet city Property:opacity\$ chart: ("#chart").ej Chart = for Chart({ new series: [{ Chart({ series: [series: [opacity: 0.7 { }];}); opacity: 0.7 }];}); </pre> <p>Property:outLierS</p> <pre> Outli ettings\$("#char er t").ejChart({ setti series: [{ ngs outLierSettin Not of gs: { shape: Applicable serie 'rectangle' , s size: { height: 30, width: 20}} }];}); </pre> <p>Property:po intColorMa ppinglet</p> <pre> Palet Property:palette\$ chart: te ("#chart").ej Chart = Chart({ new series: [{ Chart({ palette: series: ["ColorFieldNa pointColo me" }];}); rMapping: 'color' }];});cha rt.append To('#char t); </pre> <p>Property:po intColorMa ppinglet</p> <pre> Posit Property:positiveF Property:po ive ill\$("#chart"). intColorMa fill ejChart({ ppinglet for series: [{ chart: water positiveFill: Chart = "red" }];}); new rfall Chart({ </pre>
--	--	--	--	---

				<pre> series s </pre>	<pre> series: [{ pointColorMapping: 'color' }];});chart.append To('#chart'); </pre>
				<pre> Show w aver age value in box and whisker series s </pre>	<pre> Property:pointColorMappinglet chart: Chart = new Chart({ series: [{ showMean: false }];});chart.append To('#chart'); </pre>
				<pre> To group p the series of stacking collection . </pre>	<pre> Property:stackingGrouplet chart: Chart = new Chart({ series: [{ stackingGroup: 'group' }];});chart.append To('#chart'); </pre>
				<pre> Specifies the type of the series s to rend </pre>	<pre> Property:typelet chart: Chart = new Chart({ series: [{ type: 'Line' }];});chart.append </pre>

				<pre> er in char t. Property:vis iblelet chart: Chart = new the Property:visibility visibi \$("#chart").e lity jChart({ of series: [{ the visibility: serie true }]);}); s. chart.append To('#char t); Property:to ogleVisibilit ylet chart: Chart = new the Property:visibleO visibi nLegend lity \$("#chart").e of jChart({ lege series: [{ nd visibleOnLege item nd : true . }]);}); chart.append To('#char t); Property:sp lineTypele t chart: Chart = new the Property:splineTy diffe pe rent \$("#chart").e type jChart({ s of series: [{ splin splineType : e 'Natural' curve }]);}); chart.append To('#char t); </pre>
--	--	--	--	---

				<p>Specifies the name of the x-axis that has to be associated with this series.</p> <p>Add an axis instance with this name to axes collection.</p> <p>Name of the property in the data source that contains</p>	<pre> Property:xA xisName let chart: Chart = new Chart({ series: [{ xAxisName : 'secondar yXAxis' }];});cha rt.append To('#char t'); </pre> <pre> Property:xA xisName let chart: Chart = new Chart({ series: [{ xAxisName : 'secondar yXAxis' }];});cha rt.append To('#char t'); </pre>
				<p>Name of the property in the data source that contains</p>	<pre> Property:xN ame let chart: Chart = new Chart({ series: [{ xName: 'x' }];});cha rt.append To('#char t'); </pre> <pre> Property:xN ame let chart: Chart = new Chart({ series: [{ xName: 'x' }];});cha rt.append To('#char t'); </pre>

				<p>x valu e for the serie s.</p> <p>Spec ifies the nam e of the y- axis that has to be asso ciate d with this serie s.</p> <p>Add an axis insta nce with this nam e to axes colle ction .</p> <p>Nam e of the prop erty in the</p>	<p>Property:yA xisNamele t chart: Chart = new Chart({ series: [{ yAxisName : 'secondar yYAxis' }]);});cha rt.append To('#char t');</p> <p>Property:yAxisNa me\$("#chart") .ejChart({ series: [{ yAxisName : 'secondaryYAx is' }]);});</p> <p>Property:yName\$ ("#chart").ej Chart({ series: [{ yName : 'y' }]);});</p> <p>Property:yN amelet chart: Chart = new Chart({ series: [{ yName:</p>
--	--	--	--	--	---

				<pre> data sour ce that cont ains y valu e for the serie s. Nam e of the prop erty in the data sour ce that cont ains high valu e for the serie s. Nam e of the prop erty in the data sour ce that cont ains low </pre>	<pre> 'y' });});cha rt.append To('#char t'); Property:hi ghlet chart: Chart = new Chart({ series: [{ high: 'y' }]);});cha rt.append To('#char t'); Property:lo wlet chart: Chart = new Chart({ series: [{ low: 'y' }]);});cha rt.append To('#char t'); </pre>
--	--	--	--	---	---

				<div><div>valu e for the serie s.</div><div>Nam e of the prop erty in the data sour ce that cont ains close valu e for the serie s.</div><div>Nam e of the prop erty in the data sour ce that cont ains open valu e for the serie s.</div></div> <div><div>Property:close\$("#chart").ejC hart({ series: [{ close : 'y' }];});</div><div>Property:cl oselet chart: Chart = new Chart({ series: [{ close: 'y' }];});cha rt.append To('#char t');</div><div>Property:op enlet chart: Chart = new Chart({ series: [{ open: 'y' }];});cha rt.append To('#char t');</div></div>
--	--	--	--	---

				<pre> Property:trendLines t chart: Chart = new es\$("#chart").Chart({ ejChart({ series: [{ trendLines : trendLines : [{}] }];});chart.append To('#chart'); Options for customizing the appearance of the series or data point while highlighting. Options for customizing the appearance of the series </pre>
--	--	--	--	---

				<pre> s/da ta poin t on selec tion. ## marker Property:visiblelet chart: Chart = new Property:visible\$("#chart").ejChart({ series: [{ marker: { visible: true } }]}); Property:visiblelet chart: Chart = new Property:fill\$("#chart").ejChart({ series: [{ marker: { fill: 'red' } }]}); Property:opacitylet chart: Chart = new Property:opacity\$("#chart").ejChart({ series: [{ marker: { opacity: 0.5 } }]}); </pre>
--	--	--	--	--

						: 0.5 } }};});ch art.appe ndTo('#c hart);
						Property:s hapelet chart: Chart = new
				Shape of marker	Property:shape\$(" #chart").ejCha rt({ series: [{ marker: { shape : 'Circle' } }]);};	Chart({ series: [{ marker: { shape : 'Triangl e' } }]);});ch art.appe ndTo('#c hart);
						Property:i mageUrlle t chart: Chart = new
				Image Url of marker	Property:imageUrl \$("#chart").ej Chart({ series: [{ marker: { imageUrl : '' } }]);};	Chart({ series: [{ marker: { imageUrl : '' } }]);});ch art.appe ndTo('#c hart);
						Property:s hapelet chart: Chart = new
				Border of marker	Property:border\$(" #chart").ejCh art({ series: [{ marker: { border : { width: 2, color: 'red' } } }]);};	chart: Chart = new Chart({ series: [{ marker: { border : width: 2,

				<pre> color: 'red' } }]];});ch art.appe ndTo('#c hart); Property:h eightlet chart: Chart = new Property:size.height t\$("#chart").e Chart({ jChart({ series: [{ marker: { size: { height: 30} } }]];}); series: [{ marker: { height: 25 } }]];});ch art.appe ndTo('#c hart); Property:w idthlet chart: Chart = new Property:size.width h\$("#chart").e Chart({ jChart({ series: [{ marker: { size: { width: { width: 30} } }]];}); series: [{ marker: { width: 25 } }]];});ch art.appe ndTo('#c hart); Property:w idthlet chart: Chart = new Property:size.width h\$("#chart").e Chart({ jChart({ series: [{ marker: { size: { width: 30} } }]];}); series: [{ marker: { width: 25 } }]];});ch art.appe </pre>
--	--	--	--	--

				<pre> ndTo('#c hart); Property: marker.dat aLabellet chart: Chart = new Chart({ series: [{ marker: { dataLabe l: { } } }];});ch art.appe ndTo('#c hart); Property:d ataLabel.vi siblelet chart: Chart = new Chart({ series: [{ marker: { dataLabe l: { visible: true } } }];});ch art.appe ndTo('#c hart); Property:d ataLabel.n amelet chart: Chart = new Chart({ series: [{ marker: { textMappingNam e: ' ' } } }];}); dataLabe l: { </pre>
				<pre> Data Property:marker.d Label ataLabel\$("#char ISett t").ejChart({ ings series: [{ of marker: marker {dataLabel: { ker } }]]); dataLabe l: { } } }];});ch art.appe ndTo('#c hart); Property:d ataLabel.vi siblelet chart: Chart = new Chart({ series: [{ marker: { dataLabe l: { visible: true } } }];});ch art.appe ndTo('#c hart); Property:d ataLabel.n amelet chart: Chart = new Chart({ series: [{ marker: { textMappingNam e: ' ' } } }];}); dataLabe l: { </pre>

				<pre> name: '' } } }};});ch art.append ndTo('#c hart); Property:da taLabel.fil llet chart: Chart = new Property:dataLabel Fill .fill\$("#chart"). color ejChart({ r of series: [{ marker: data {dataLabel: { label fill: 'pink' } } }];}); Property:da taLabel.fil llet chart: Chart = new Property:dataLabel Opa .opacity\$("#char city t").ejChart({ of series: [{ data marker: label {dataLabel: { opacity: 0.6 } } }];}); Property:da taLabel.p ositionlet Text .textPosition\$("#c posit hart").ejChart ion ({ series: [{ of marker: {dataLabel: { </pre>
--	--	--	--	---

				<pre> data textPosition: Chart({ label 'middle' } } series:]);}); marker: { dataLabe l: { position : 'Top' } } });});ch art.appe ndTo('#c hart); Property:d ataLabel.al ignmentle t chart: Chart = new Property:dataLabel .verticalAlignment Align \$("#chart").ej men Chart({ t of series: [{ data marker: label {dataLabel: { verticalAlignm dataLabe ent: 'near' } l: { } }]);}); alignment t: 'Near' } } });});ch art.appe ndTo('#c hart); Property:d ataLabel.al ignmentle t chart: Chart = new Property:dataLabel .border\$("#char t").ejChart({ Bord series: [{ er of marker: data {dataLabel: { label border: { color: 'blue', width: 2, opacity: 0.4} } } }]);}); border: { color: 'blue', width: 2 </pre>
--	--	--	--	--

				<pre> } } } }]);});ch art.append ndTo('#c hart); </pre>
				<pre> Property:dataLabel .offset\$("#chart ").ejChart({ series: [{ marker: {dataLabel: { offset: { x: 5, y: 6 }} } }]);}); </pre>
				<pre> Property:d ataLabel.m arginlet chart: Chart = new Chart({ series: [{ marker: { dataLabe l: { margin: { top: { top: 10, bottom: 10, left: 10, right: 10 } } } }]);});ch art.append ndTo('#c hart); </pre>
				<pre> Property:dataLabel .border\$("#char t").ejChart({ series: [{ marker: {dataLabel: { margin: { top: 10, bottom: 10, left: 10, right: 10}} } }]);}); </pre>
				<pre> Property:d ataLabel.m arginlet chart: Chart = new Chart({ series: [{ marker: {dataLab font: { fontFamily: 'SegoeUI', fontStyle: 'italic', </pre>

					fontWeight: el: { '600', font: { opacity: 0.5, fontFami size: 12, ly: color: 'red' 'SegoeUI }} } }};});', fontStyl e: 'italic' , fontWeig ht: '600', opacity: 0.5, size: 12, color: 'red' }}} }};});ch art.appe ndTo('#c hart);	
					Property:d ataLabel.te mplatelet chart: Chart = new Chart({ series: [{ marker: {dataLab el: { template : ' Chart ' } } }];});ch art.appe ndTo('#c hart);	
				HTM L tem plat e in data Labe l	Property:dataLabel .template\$("#cha rt").ejChart({ series: [{ marker: {dataLabel: { template: ' Chart ' } } }];});});	
				Rou nde d corn er	Not Applicable Property:d ataLabel.rx let chart: Chart = new Chart({	

				<pre> radi us X series: [{ marker: {dataLab el: { rx: 10 } } }];});ch art.appe ndTo('#c hart); Property:d ataLabel.ry let chart: Chart = new Chart({ series: [{ marker: {dataLab el: { ry: 10 } } }];});ch art.appe ndTo('#c hart); Rou nde d corn Not Applicable er radi us Y Property: dataLabel.ry let chart: Chart = new Chart({ series: [{ marker: {dataLab el: { ry: 10 } } }];});ch art.appe ndTo('#c hart); Maxi Property: mu .maximumLabelWi m dth\$("#chart") Labe .ejChart({ series: [{ marker: width {dataLabel: { h for maximumLabelWi data dth: 20}} } label }];}); Enab Property: le .enableWrap\$("# wra chart").ejChar ppin t({ series: [{ g of marker: text {dataLabel: { for enableWrap: data true }} } label }];}); </pre>
--	--	--	--	---

				<pre> To Property:dataLabel sho .showContrastColo w r\$("#chart").e cont jChart({ rast series: [{ Not colo marker: Applicable r for {dataLabel: { data showContrastCo label lor: true }} } }];}); To Property:dataLabel sho .showEdgeLabels\$ w (\$("#chart").ejC edge hart({ series: Not label [{ marker: Applicable for {dataLabel: { data showEdgeLabels : true }} } label }];}); ## TrendLines Be ha API in Essential JS API in Essential vi 1 JS 2 ou r Tr Property:series en .trendLineslet dli Property:series.tr chart: ne endLines\$("#ch Chart = new s art").ejChart Chart({ se ({ series: [{ series: [{ tti trendLines: trendLines: ng []}}]); []}}]);char s t.appendTo('#chart'); Vis ibi Property:trendLi lit nes.visibility\$ (" y #chart").ejCh of art({ series: Not applicable tre [{ nd trendLines: [lin visibility: e true]}}]); </pre>
--	--	--	--	---

				<pre> Property:trend Lines.typelet chart: nes.type\$("#ch Chart = new art").ejChart Chart({ series: [{ series: [{ trendLines: trendLines: [type: [type: 'linear' 'Polynomial }]}}]); ']}}]);char t.appendTo('#chart'); </pre>
				<pre> Property:trend Lines.namelet t chart: nes.name\$("#c Chart = new hart").ejChar Chart({ series: [{ trendLines: [name: 'trendLine']}]);chart .appendTo('# chart'); </pre>
				<pre> Property:trend Lines.periodlet t chart: nes.period\$("#c Chart = new hart").ejChar Chart({ series: [{ trendLines: [period: 45]}]);chart .appendTo('#chart'); </pre>
				<pre> Property:trend Lines.polynomi alOrderlet chart: nes.polynomialO Chart = new rder\$("#chart Chart({ series: [{ trendLines: [polynomialOrd polynomialO er: 3]}}]); rder: 3]}}]);char t.appendTo('#chart'); </pre>

				<pre> ty pe tre nd Lin es Ba ck w ar d for ec os t for tre nd Lin es Fo rw ar d for ec os t for tre nd Lin es Fill for tre nd Lin es </pre>
				<pre> Property:trend Lines.backwardforecastlet chart: Chart = new Chart({ series: [{ trendLines: [[backwardforecast backwardfor ast: 3]]] }]; ecast: 3]]] }]; char t.appendTo('#chart'); Property:trend Lines.forwardForecastlet chart: Chart = new Chart({ series: [{ trendLines: [[forwardForecast forwardFore st: 3]]] }]; cast: 3]]] }]; char t.appendTo('#chart'); Property:trend Lines.filllet chart: Chart = new Chart({ series: [{ trendLines: [trendLines: [fill: ['EEFFCC' 'EEFFCC']]]]] }]; });chart.ap pendTo('#ch art'); </pre>

				<pre> Property:trend Lines.widthle t chart: Chart = new Chart({ series: [{ trendLines: [width: 2]]}});char t.appendTo('#chart'); </pre>
				<pre> Property:trend Lines.intercept let chart: Chart = new Chart({ series: [{ trendLines: [intercept: 2]]}});char t.appendTo('#chart'); </pre>
				<pre> Property:trend Lines.legendSh apelet chart: Chart = new Chart({ series: [{ trendLines: [legendShape : 'Rectangle']]}});chart .appendTo('#chart'); </pre>
				<pre> Property:trend Lines.animation let chart: Chart = new Chart({ series: [{ trendLines: [animation: { enable: true }}]]});char </pre>

				<pre> nd Lin es M ar ke r se tti ng Not Applicable s for tre nd Lin es Property:trend Lines.marker1 et chart: Chart = new Chart({ series: [{ trendLines: [{marker: { visible: true }}]}]);cha rt.appendTo ('#chart'); Property:trend Lines.enableTo oltiplet chart: Chart = new Chart({ series: [{ trendLines: [{ trendLines: [[{ tooltip: { {enableTool } }]}}]); tip: true }}]}]);char t.appendTo('#chart'); Da sh Property:trendLi Ar nes.dashArray\$(ra "#chart").ejC y hart({ Not for series: [{ Applicable. tre trendLines: [nd { dashArray: Lin '10, 5' } es]}}]); Vis Property:trendLi ibl nes.visibleOnLeg e end\$("#chart" Not on).ejChart({ Applicable. le series: [{ trendLines: [</pre>
--	--	--	--	--

				<pre> ge { nd visibleOnLege nd: true } for]}}}); tre nd Lin es ## StripLines Be hav API in Essential API in Essential iou JS 1 JS 2 r Property:prima ryXAxis.stripLi neslet aul Property:primar chart: t yXAxis.stripLines Chart = new be \$("#chart"). Chart({ hav ejChart({ primaryXAxis iou primaryXAxis s: { r : { stripLines: for stripLines: [{ stri [{ visible: visible: pli true }]]}); true nes .appendTo('# chart'); Property:stripL ines.borderColor Property:stripLin t chart: es.borderColor\$ Chart = new bor (\$("#chart").e Chart({ der jChart({ primaryXAxis for primaryXAxis s: { : { stripLines: stri stripLines: [{ border: pli [{ { color: ne borderColor: 'red', 'pink' width: 2} }}}}); }]]});chart .appendTo('# chart'); Bac Property:stripLin Property:stripL kgr es.color\$("#ch ines.borderColor ou art").ejChar t chart: nd t({ Chart = new col primaryXAxis Chart({ </pre>
--	--	--	--	---

				<pre> or : { for stripLines: s: { stri [{ color: stripLines: pli 'pink' [{ color: ne }}}]); 'red'}}]); chart.append dTo('#chart '); Property:stripL ines.startlet chart: es.start\$("#ch rt").ejChar val Chart = new ue Chart({ for primaryXAxis primaryXAxis : { s: { stri stripLines: stripLines: pli [{ start: [{ start: ne 10 }}}]); 5}}]);char t.appendTo('#chart'); Property:stripL ines.endlet chart: es.end\$("#cha rt").ejChart val Chart = new ue Chart({ for primaryXAxis primaryXAxis : { s: { stri stripLines: stripLines: pli [{ end: 10 [{ end: ne }}}]); 5}}]);char t.appendTo('#chart'); Property:stripL ines.startFrom Axislet chart: es.startFromAxis rtfr \$("#chart"). om ejChart({ Axi primaryXAxis primaryXAxis s: { s: { for stripLines: stripLines: stri [{ startFromAx pli s: true is: ne }}}]); true}}]);c hart.append To('#chart '); Tex Property:stripL Property:stripL t in es.text\$("#cha ines.textlet </pre>
--	--	--	--	--

				<pre> stri rt").ejChart chart: pli ({ Chart = new ne primaryXAxis Chart({ : { primaryXAxis stripLines: s: { [{ text: stripLines: 'StripLine; [{ text: }}]); 'stripline' }}]); }]);chart .appendTo('#chart'); Property:stripL ines.horizontal Tex Property:stripLin Alignmentlet t es.textAlignment chart: alig t\$("#chart") Chart = new nm .ejChart({ Chart({ ent primaryXAxis s: { in stripLines: stripLines: stri [{ horizontalA pli textAlignmen lignment: ne t: 'Far; 'Far'}}]); chart.appen dTo('#chart '); Property:stripL ines.verticalAli gnmentlet chart: Chart = new Chart({ primaryXAxis s: { stripLines: [{ verticalAli gnment: 'Far'}}]); chart.appen dTo('#chart '); Property:stripLin es.width\$("#ch art").ejChar t({ primaryXAxis : { stripLines: stripLines: </pre>
--	--	--	--	--

					<pre> [{ width: [{ size: 10; }]]]); 10 }}]);chart .appendTo('#chart'); </pre>
					<p>Property:stripLines.size</p> <pre> let chart: es.zIndex\$("#chart").ejChart({ primaryXAxis: { stripLines: [{ zIndex: 'Behind' }]]]);chart .appendTo('#chart'); </pre>
					<p>Property:stripLines.textStyle</p> <pre> let chart: es.fontStyle\$("#chart").ejChart({ primaryXAxis: { stripLines: [{ fontStyle: {} }]]]);chart .appendTo('#chart'); </pre>
					<h3>## Multilevel Labels</h3> <p>Be hav iou r</p> <p>API in Essential JS API in Essential JS 2</p>
					<pre> Defaul t be hav iou r for mu ltile Property:primaryXAxis.multilevelLabels\$("#chart").ejChart({ primaryXAxis: { multilevelLabels: [{ visible: true }]]]); </pre>
					<p>Property:primaryXAxis.multilevelLabels</p> <pre> let chart: Chart = new Chart({ primaryXAxis: { is: { multilevelLabels: [{ </pre>

				<pre> vel }}}});char Lab t.appendTo els ('#chart') ; Def Property:prim aul aryXAxis.muti t Property:primary levelLabelsle be XAxis.multilevelL t chart: hav abels\$("#chart new iou ").ejChart({ Chart({ r primaryXAxis: primaryXAx { is: { for multilevelLab multilevel mu els: [{ Labels: [ltile visible: true { vel }}}}); }]);char Lab t.appendTo els ('#chart') ; Property:mult ilevelLabels.ali Tex gnmentlet t Property:multiLe chart: alig vellLabels.textAlig Chart = nm nment\$("#char new ent t").ejChart({ Chart({ for primaryXAxis: primaryXAx { is: { mu multilevelLab multilevel ltile els: [{ Labels: [vel textAlignment {alignment Lab : 'Near' : 'Near' els }}}}); }]);char t.appendTo ('#chart') ; Property:mult ilevelLabels.o Tex Property:multiLe t vellLabels.textOv verFlowlet ove erFlow\$("#cha chart: rflo rt").ejChart(Chart = w { new for primaryXAxis: Chart({ mu { primaryXAx ltile multilevelLab is: { vel els: [{ multilevel Lab textOverFlow: Labels: [els 'Trim' }]); }overFlow: 'Trim' }}]);char </pre>
--	--	--	--	---

				<pre> t.appendTo ('#chart') ; Property:multiLevelLabels.b orderlet chart: Chart = new \$("#chart").ejChart({ primaryXAxis: is: { multilevel Labels: [els: [{ border: { width: 2, color: 'red', type: 'brace' } }]]]); t.appendTo ('#chart') ; Property:multiLevelLabels.c ategories.start tlet chart: Chart = new \$("#chart").ejChart({ primaryXAxis: is: { multilevel Labels: [categories start: 45 } : [{ start: 45}] }]]]]); t.appendTo ('#chart') ; Property:multiLevelLabels.c ategories.end let chart: Chart = new \$("#chart").ejChart({ primaryXAxis: is: { multilevel Labels: [categories start: 45 } : [{ start: 45}] }]]]]); t.appendTo ('#chart') ; </pre>
--	--	--	--	--

				<pre> lab els: [{ end: primaryXAx el 45 }]]]]]); is: { multilevel Labels: [{ categories : [{ end: 45}] }]]]]); char t.appendTo ('#chart') ; Property:mult iLevelLabels.c ategories.text let chart: Chart = new Property:multile velLabels.text\$("#chart").ejC Tex t hart({ for primaryXAxis: multilevel lab multilevelLab Labels: [el els: [{ categories : [{ text: 'Start' text: 'text' }] }]]]]); char t.appendTo ('#chart') ; Property:mult iLevelLabels.c ategories.max Property:multile velLabels.maxim umTextWidth\$("#chart").ejC Tex t hart({ wid primaryXAxis: Chart({ th multilevelLab is: { for els: [{ multilevel lab maximumTextWi Labels: [el dth: 10 } { categories : [{ maximumTex tWidth: 20 }] } </pre>
--	--	--	--	--

				<pre> }}}});char t.appendTo ('#chart') ; Property:multile vellLabels.level\$(lev "#chart").ejC el hart({ of primaryXAxis: Not { applicable. lab multilevelLab Categories are els els: [{ used level: 2 } }}}}); ## Methods Be ha API in API in Essential JS vio Essential JS 1 2 ur ani Property:cha ma rt.animate\$(tio "#chart" n).ejChart Not applicable for ({ ser animate: ies () { }); Re Property:cha Property:chart.refr dra rt.redraw\$(esh())let chart: w "#chart") Chart = new for .ejChart(Chart({});char cha { redraw: t.appendTo('#c rt () { }); hart');chart.w idth = '400';chart.re fresh(); Property:chart.exp Property:cha ort()let chart: rt.export\$(Chart = new Exp "#chart") Chart({});char ort .ejChart(t.export('JPEG { export: ', () { }); 'chart');chart .appendTo('#ch art'); </pre>
--	--	--	--	--

				<pre> Property:chart.print()\$("#chart").ejChart({ print: () { {} } }); </pre>	<pre> Property:chart.print()let chart: Chart = new Chart({});chart t.print('chart ');chart.append dTo('#chart'); </pre>
				<pre> AddSeries()let chart: Chart = new Chart({});chart t.appendTo('#c hart');chart.a ddSeries(); </pre>	<pre> Property:chart.add Series()let chart: Chart = new Chart({});chart t.appendTo('#c hart');chart.a ddSeries(); </pre>
				<pre> RemoveSeries()let chart: Chart = new Chart({});chart t.appendTo('#c hart');chart.r emoveSeries(); </pre>	<pre> Property:chart.re moveSeries()let chart: Chart = new Chart({});chart t.appendTo('#c hart');chart.r emoveSeries(); </pre>
				## Events	
				<pre> Behavior: API in API in Essential JS 1 Essential JS 2 </pre>	<pre> API in Essential JS 2 </pre>
				<pre> Fire on annotationClick\$("#chart").ejChart({ annotationClick: () { {} } }); </pre>	<pre> Property:annotationClick\$("#chart").ejChart({ annotationClick: () { {} } }); </pre>
				<pre> Fire afterAnimationComplete\$("#chart").ejChart({ animationComplete: () { {} } }); </pre>	<pre> Property:animationComplete\$("#chart").ejChart({ animationComplete: () { {} } }); </pre>

				<pre> { });char t.append To('#cha rt'); </pre> <p>Fire</p> <p>s</p> <p>on Property:axisLabelClick</p> <p>axisLabelClick</p> <p>el : () { });</p> <p>Not applicable</p> <p>Property:axisLabelRender</p> <p>Fire</p> <p>s</p> <p>before Property:axisLabelRender</p> <p>axisLabelRender</p> <p>el : () { });</p> <p>Not applicable</p> <p>Property:axisLabelMouseMove</p> <p>Fire</p> <p>s</p> <p>on Property:axisLabelMouseMove</p> <p>axisLabelMouseMove</p> <p>el : () { });</p> <p>Not applicable</p> <p>Property:axisLabelInitialize</p> <p>Fire</p> <p>s</p> <p>on Property:axisLabelInitialize</p> <p>axisLabelInitialize</p> <p>el : () { });</p> <p>Not applicable</p>
--	--	--	--	---

				<pre> Fire s bef ore Property:axesRangeCalculated()let axis eCalculate\$("#chart").ejChart({ ran axesRangeCalculate: () { ge late: () { calc }}); ula tio n </pre> <p>Property:axisRangeCalculated()let chart: Chart = new Chart({ axisRangeCalculate: late: () { ted: () => { }});chart. append To('#chart');</p> <pre> Fire s on Property:axisTitleRendering\$ axis t").ejChart({ titl axisTitleRendering: () { e ring: () { ren }}); der ing </pre> <p>Not applicable</p> <pre> Fire s on Property:afterResize\$ aft e\$("#chart").ejChart({ er jChart({ cha afterResize: rt () { }}); resi ze </pre> <p>Not applicable</p> <pre> Fire s on Property:beforeResize\$ bef ize\$("#chart").ejChart({ ore ejChart({ cha beforeResize: rt () { }}); resi ze </pre> <p>Property:resizedlet chart: Chart = new Chart({ resized: () => { }});chart. append To('#chart');</p> <pre> Fire Property:chartClick\$ s \$("#chart").ej </pre> <p>Property:chartMouseClick</p>
--	--	--	--	---

				<pre> on Chart({ cha chartClick: () rt { }); clie ck icklet chart: Chart = new Chart({ chartMou seClick: () => { }});char t.append To('#cha rt'); Property:ch artMouseM ovelet chart: Chart = new Chart({ chartMou seMove: () => { }});char t.append To('#cha rt'); Property:ch artMouseLe avelet chart: Chart = new Chart({ chartMou seLeave: () => { }});char t.append To('#cha rt'); Fire s on Property:chartDou bleClick\$("#chart ").ejChart({ chartDoubleCli ck: () { }); do ubl </pre>
--	--	--	--	--

				<p>click</p> <p>Not Applicable</p> <p>Property:chartmouseUp</p> <pre> plet chart: Chart = new Chart({ chartmouseUp: () => { }});chart t.append To('#chart'); </pre> <p>Property:chartmouseDown</p> <pre> ownlet chart: Chart = new Chart({ chartmouseDown: () => { }});chart t.append To('#chart'); </pre> <p>Not applicable</p> <p>Property:chartArea</p> <pre> BoundsCalculate\$("#chart").ejChart({ chartAreaBoundsCalculate: () { }}); </pre> <p>Not applicable</p>
--	--	--	--	--

				<p>You can use this event to customize the bounds of chart area</p> <p>Fire when the dragging started</p> <pre>Property:dragStart \$("#chart").ejChart({ dragStart: () { { } } });</pre> <p>is started</p> <p>Fire when the dragging started</p> <pre>Property:dragging\$ \$("#chart").ejChart({ dragging: () { { } } });</pre> <p>is started</p> <p>Fire when the dragging completed</p> <pre>Property:dragEnd\$ \$("#chart").ejChart({ dragEnd: () { { } } });</pre> <p>is started</p> <p>Property:dragComplete</p> <pre>let chart: Chart = new Chart({</pre>
--	--	--	--	---

				<pre> ggi ng is co mpl ete d Fire s wh en cha rt is des tro yed co mpl etel y. Fire s aft er cha rt is cre ate d. Fire s bef ore ren der ing the dat a lab els. </pre>	<pre> dragComp lete: () => { }});char t.append To('#cha rt'); Property:destroy\$("#chart").ejCh Not art({ destroy: applicable () { }}); Property:lo adedlet chart: Chart = new Chart({ loaded: () => { }});char t.append To('#cha rt'); Property:te xtRenderle t chart: Chart = new Chart({ textRend er: () => { }});char t.append To('#cha rt'); </pre>
--	--	--	--	---	---

				<pre> Fire s, wh en Property:errorBarR err endering\$("#char Not or t").ejChart({ applicable bar errorBarRender is ing: () { }); ren der ing. Fire s dur ing the Property:legendBo calc undsCalculate\$("# ula chart").ejChar Not tio t({ applicable n legendBoundsCa of lculate: () { leg }); end bo un ds. Fire s on clic Property:legendIte kin mClick\$("#chart Not g ").ejChart({ applicable the legendItemClick leg k: () { }); end ite m. Fire Property:legendIte s mMouseMove\$("# wh #chart").ejCha Not en rt({ applicable mo legendItemMous vin eMove: () { g }); </pre>
--	--	--	--	---

				<pre> mo use ove r leg end ite m Fire s bef ore Property:legendItemRendering\$ mRendering\$("#chart").ejChart ({ legendItemRendering: () { }}}); Property:legendRenderlet chart: Chart = new Chart({ legendRender: () => { }}});chart t.append To('#chart'); Fire s bef ore Property:load\$ ("#chart").ejChart t({ load: () { }}}); Property:loadadlet chart: Chart = new Chart({ load: () => { }}});chart t.append To('#chart'); Fire s, wh en Property:multiLevelLabelRendering\$ ("#chart").ejChart ({ multiLevelLabelRendering: () { }}}); Property:axisMultiLevelRenderlet chart: Chart = new Chart({ axisMultiLevelRender: () => { }}});chart t.append </pre>
--	--	--	--	--

				<pre> der ing. Fire s on clic kin g a poi nt in cha rt. Fire s wh en mo use is mo ved ove ra poi nt. Fire s bef ore ren der ing cha rt. Fire s wh en poi nt ren </pre>	<pre> To('#cha rt'); Property:po intClick let chart: Chart = new Chart({ pointCli ck : () => { }});char t.append To('#cha rt'); Property:po intMove let chart: Chart = new Chart({ pointMov e : () => { }});char t.append To('#cha rt'); Property:preRende r\$("#chart").ej Chart({ preRender: () { }}); Property:po intRender let chart: Chart = new Chart({ pointRen der : () </pre>
--	--	--	--	--	--

				<pre> der . => { });char t.append To('#cha rt'); Fire s aft er sel Property:rangeSele ect ct\$("#chart") Not ed .ejChart({ applicable the rangeSelected: dat () { }}); a in cha rt. Fire s aft er Property:seriesRegi sel onClick\$("#chart Not ecti ").ejChart({ applicable ng seriesRegionCl a ick: () { }}); seri es. Fire s bef ore Property:seriesRen ren dering\$("#chart" der).ejChart({ ing seriesRenderin a g: () { }}); seri es. Property:se riesRender let chart: Chart = new Chart({ seriesRe nder : () => { });char t.append To('#cha rt'); Fire Property:symbolRe s ndering\$("#chart Not bef ").ejChart({ applicable ore symbolRenderin ren g: () { }}); </pre>
--	--	--	--	---

				<pre>der ing the ma rke r sy mb ols. Fire s bef ore Property:trendline ren Rendering\$("#cha rt").ejChart({ Not der trendlineRende applicable ing ring: () { the }}}); tre ndli ne Fire s bef ore ren Property:titleRende der ring\$("#chart") Not ing .ejChart({ applicable the titleRendering : () { }}}); Cha rt titl e. Fire s bef ore Property:subTitleR ren endering\$("#char Not der t").ejChart({ applicable ing subTitleRender the ing: () { }}}); Cha rt sub</pre>
--	--	--	--	---

				<pre> titl e. Fire s bef ore ren der ing the too ltip . Fire s bef ore ren der ing cro ssh air too ltip in axis Fire s bef ore ren der ing tra ckb all too ltip . Eve nt </pre>	<pre> Property:to oltipRender let chart: Chart = new Chart({ tooltipR ender : () => { }});char t.append To('#cha rt'); Property:trackAxisT oolTip\$("#chart").ejChart({ trackAxisToolT ip: () { }}); Property:trackTool Tip\$("#chart"). ejChart({ trackToolTip: () { }}); Property:scrollStart \$("#chart").ej Chart({ </pre>	<pre> Not applicable Not applicable </pre>
--	--	--	--	---	---	---

				<pre> trig scrollStart: let ger () { }); chart: ed Chart = wh new en Chart({ scr scrollSt oll art : () star => { ts. });char t.append To('#cha rt'); Event Property:sc nt rollEndlet trig chart: ger Property:scrollEnd\$ Chart = ed ("#chart").ejC new wh hart({ Chart({ en scrollEnd: () d: () => scr { }); { oll });char end t.append s. To('#cha rt'); Event Property:sc nt rollChange trig let ger chart: ed Property:scrollChan Chart = wh ge\$("#chart").e new en jChart({ Chart({ scr scrollChange: scrollCh oll ange: () cha => { nge });char s. t.append To('#cha rt'); Fire Property:zo s omComple whi elet le Property:zoomCom chart: per plete\$("#chart" Chart = for).ejChart({ new min zoomComplete: Chart({ g () { }); zoomComp rec lete: () tan => { gle });char t.append </pre>
--	--	--	--	---

				<pre> zooming:\$("#chart").ejChart({ zooming: { enable: true, enableDeferedZoom: true, enablePinch: true, enableMouseWheel: true, enableScrollBar: To('#chart'); ## Chart properties Behaviour: API in Essential JS 1 API in Essential JS 2 Property:selectedDataPointIndexes:\$("#chart").ejChart({ selectedDataPointIndexes: [{ seriesIndex: 0, x: 0, pointIndex: 1}]); chart.appendTo('#chart'); Property:select edDataIndexes:\$("#chart").ejChart({ selectedDataPointIndexes: [{ seriesIndex: 0, x: 0, pointIndex: 1}]); chart.appendTo('#chart'); Property:sideBySideSeriesPlacement:\$("#chart").ejChart({ sideBySideSeriesPlacement: true}); chart.appendTo('#chart'); Property:zooming:\$("#chart").ejChart({ zooming: { enable: true, enableDeferedZoom: true, enablePinch: true, enableMouseWheel: true, enableScrollBar: let chart: Chart = new Chart({ sideBySidePlacement: true}); chart.appendTo('#chart'); let chart: Chart = new Chart({ zoomSetting s: { enable: true, enablePinch Zooming: true, enableDeffe redZooming: true enableMouse WheelZoomin </pre>
--	--	--	--	--

				<pre> true, g: true, toolBarItem enableSelec ms: [], tionZooming type: 'X' : true, }); enableScrol lBar: true });chart.a ppendTo('#c hart'); </pre>
Back grou nd color of the chart	Property:back ground \$("#contai ner").ejCh art({ background : 'transparent' });	Property:back groundlet chart: Chart = new Chart({ background: '#EEFFCC'}) ;chart.appe ndTo('#char t');		
URL of the imag e to be used as chart backg round d.	Property:back groundImage Url \$("#contai ner").ejCh art({ backGround : ImageUrl : '../images /chart/whe at.png'});	Not Applicable		
Custo mizin g bord er of the chart	Property:bor der \$("#contai ner").ejCh art({ border: { width: 2, color: '#CCEEFF', opacity: 0.5}});	Property:borde rlet chart: Chart = new Chart({ border: { width: 2, color: '#CCEEFF'}) ;chart.app endTo('#cha rt');		
This provi des optio ns for custo	Property:exp ortSettings\$("#containe r").ejChar t({ exportSett	Property:export ()let chart: Chart = new Chart({ border: { width: 2,		

				<pre>mizing filename : '#CCEEFF'}) g "chart",);chart.app export angle: endTo('#cha t '45' })); rt');chart. setting export(type gs , fileName); Property:chartA Property:chartArea let chart: tArea\$("#container"). Chart = new ejChart({ Chart({ chartArea: chartArea: { { background background: : 'transparent' custo 'transparent', border: mizant', { width: 2, ion border: { color: opacity: '#CCEEFF' 0.3, });chart.a color: ppendTo('#c 'red', hart');char width: t.export(ty 2}}}); pe, fileName);</pre>
--	--	--	--	---

```
<tr>
<td><b>Behaviour</b></td>
<td><b>API in Essential JS 1</b></td>
<td><b>API in Essential JS 2</b></td>
</tr>
<tr>
<td><b>crosshair</b></td>
<td>
<b>Property</b><i>visible</i>
</br>
</br>
<code>
$("#container").ejChart({
crosshair: { visible: true}
```

```

});
</code>
</td>
<td>
<b>Property</b>:<i>enable</i>
<br>
<br>
<code>
let chart: Chart = new Chart({
crosshair: { enable: false }
});
chart.appendTo('#chart');
</code></td>
</tr>
<tr>
<td><b>trackballTooltipSettings</b></td>
<td>
<b>Property</b>:<i>trackballTooltipSettings</i>
<br>
<br>
<code>
$("#container").ejChart({
crosshair : { trackballTooltipSettings : { border : { width :2 } } }
});
</code>
</td>
<td>
Not applicable
</td>
</tr>
<tr>
<td><b>marker</b></td>
<td>

```


Property:*marker*

```
$("#container").ejChart({
  crosshair : { marker : { border : { width :2 } } }
});
```

Not applicable

crosshair line style

Property:*line*

```
$("#container").ejChart({
  crosshair : { line: { width: 2, color: 'red' } }
});
```

```
let chart: Chart = new Chart({
  crosshair: { border: { width: 2, color: 'black' } }
});
```

type

```

<td>
<b>Property</b>:<i>type</i>
</br>
</br>
<code>
$("#container").ejChart({
crosshair : { type: 'trackball' }
});
</code>
</td>

```

```

<td>
Not applicable
</td>
</tr>

```

3D chart

<!-- markdownlint-disable MD033 -->

Behaviour	API in Essential JS 1	API in Essential JS 2
3d chart	Property:enable3D\$ ("#container").ejChart({ enable3D: true});	Not applicable
Rotation of 3d chart	Property:enableRotation\$ ("#container").ejChart({ enableRotation: false});	Not applicable
depth	Property:depth\$ ("#container").ejChart({ depth: 45});	Not applicable

Canvas rendering

<!-- markdownlint-disable MD033 -->

Behaviour	API in Essential JS 1	API in Essential JS 2
canvas rendering	Property:enableCanvasRendering\$ ("#container").ejChart({ enableCanvasRendering: true});	Not applicable

Indicators

<!-- markdownlint-disable MD033 -->

Behaviour	API in Essential JS 1	API in Essential JS 2
-----------	-----------------------	-----------------------

Type of Indicator	Property:type <code>\$("#container").ejChart({ indicators: [{ type: 'Sma' }]});</code>	Property:type <code>let chart: Chart = new Chart({ indicators: [{ type: 'Sma' }]});</code>
Period for indicator	Property:period <code>\$("#container").ejChart({ indicators: [{ period: 14 }]});</code>	Property:type <code>let chart: Chart = new Chart({ indicators: [{ period: 14 }]});</code>
Period for indicator	Property:period <code>\$("#container").ejChart({ indicators: [{ period: 14 }]});</code>	Property:period <code>let chart: Chart = new Chart({ indicators: [{ period: 14 }]});</code>
%K value in stochastic indicator	Property:kPeriod <code>\$("#container").ejChart({ indicators: [{ kPeriod: 14 }]});</code>	Property:kPeriod <code>let chart: Chart = new Chart({ indicators: [{ kPeriod: 14 }]});</code>
%D value in stochastic indicator	Property:dPeriod <code>\$("#container").ejChart({ indicators: [{ dPeriod: 3 }]});</code>	Property:dPeriod <code>let chart: Chart = new Chart({ indicators: [{ dPeriod: 3 }]});</code>
Shows overSold/overBought values	Not applicable	Property:showZones <code>let chart: Chart = new Chart({ indicators: [{ showZones: true }]});</code>
Overbought value for RSI and stochastic indicator	Not applicable	Property:overBought <code>let chart: Chart = new Chart({ indicators: [{ overBought: 80 }]});</code>
Oversold value for RSI and stochastic indicator	Not applicable	Property:overSold <code>let chart: Chart = new Chart({ indicators: [{ overSold: 20 }]});</code>
Standard deviation for Bollingerbands	Property:standardDeviations <code>\$("#container").ejC hart({ indicators: [{ standardDeviations: 2 }]});</code>	Property:standardDeviation <code>1 et chart: Chart = new Chart({ indicators: [{ standardDeviation: 2 }]});</code>
standard deviation	Property:standardDeviations <code>\$("#container").ejC hart({ indicators: [{ standardDeviations: 2 }]});</code>	Property:standardDeviation <code>1 et chart: Chart = new Chart({ indicators: [{ standardDeviation: 2 }]});</code>
Field for indicator	Property:field <code>\$("#container").ejChart({ indicators: [{ field: 'Close' }]});</code>	Property:field <code>let chart: Chart = new Chart({</code>

		<code>indicators: [{ field: 'Close' }]]);</code>
Slow period for MACD indicator	<code>Property:shortPeriod\$("#container").ejChart({ indicators: [{ shortPeriod: 12 }]]);</code>	<code>Property:slowPeriodlet chart: Chart = new Chart({ indicators: [{ slowPeriod: 12 }]]);</code>
Fast period for MACD indicator	<code>Property:longPeriod\$("#container").ejChart({ indicators: [{ longPeriod: 26 }]]);</code>	<code>Property:fastPeriodlet chart: Chart = new Chart({ indicators: [{ fastPeriod: 26 }]]);</code>
Line style for MACD indicator	<code>Property:macdLine\$("#container").ejChart({ indicators: [{ macdLine: { width: 2, fill: 'red' } }]]);</code>	<code>Property:fastPeriodlet chart: Chart = new Chart({ indicators: [{ macdLine: { width: 2, color: 'red' } }]]);</code>
Type of MACD indicator	<code>Property:macdType\$("#container").ejChart({ indicators: [{ macdType: 'both' }]]);</code>	<code>Property:fastPeriodlet chart: Chart = new Chart({ indicators: [{ macdType: 'Both' }]]);</code>
Color of the positive bars in Macd indicators	Not applicable	<code>Property:macdPositiveColor let chart: Chart = new Chart({ indicators: [{ macdPositiveColor: 'red' }]]);</code>
Color of the negative bars in Macd indicators	Not applicable	<code>Property:macdNegativeColor let chart: Chart = new Chart({ indicators: [{ macdNegativeColor: 'red' }]]);</code>
Color for Bollinger bands	Not applicable	<code>Property:bandColorlet chart: Chart = new Chart({ indicators: [{ bandColor: 'red' }]]);</code>
Appearance of upper line in indicator	<code>Property:upperLine\$("#container").ejChart({ indicators: [{ upperLine: { fill: '#EECCAA', width: 2 } }]]);</code>	<code>Property:upperLinelet chart: Chart = new Chart({ indicators: [{ upperLine: { type: 'Smooth', color: '#FFEEFF', width: 2, dashArray: '10,5' } }]]);</code>

Appearance of lower line in indicator	Property:lowerLine\$("#container").ejChart({ indicators: [{ lowerLine: { fill: '#EECCAA', width: 2 } }] });	Property:lowerLinelet chart: Chart = new Chart({ indicators: [{ lowerLine: { type: 'Smooth', color: '#FFEEFF', width: 2, dashArray: '10,5' } }] });
Appearance of period line in indicator	Property:periodLine\$("#container").ejChart({ indicators: [{ periodLine: { fill: '#EECCAA', width: 2 } }] });	Property:lowerLinelet chart: Chart = new Chart({ indicators: [{ lowerLine: { type: 'Smooth', color: '#FFEEFF', width: 2, dashArray: '10,5' } }] });
Name of the series for which indicator has to be drawn.	Property:seriesName\$("#container").ejChart({ indicators: [{ seriesName: '' }] });	Property:lowerLinelet chart: Chart = new Chart({ indicators: [{ seriesName: '' }] });
Options to customize the histogram in MACD indicator	Property:seriesName\$("#container").ejChart({ indicators: [{ histogram: { } }] });	Not applicable
Enabling animation	Property:enableAnimation\$("#container").ejChart({ indicators: [{ enableAnimation: true }] });	Property:animation.enablelet chart: Chart = new Chart({ indicators: [{ animation: { enable: true } }] });
Animation duration	Property:animationDuration\$("#container").ejChart({ indicators: [{ animationDuration: 3000 }] });	Property:animation.durationlet chart: Chart = new Chart({ indicators: [{ animation: { duration: 3000 } }] });
Tooltip	Property:tooltip\$("#container").ejChart({ indicators: [{ tooltip: { visible: true } }] });	Not applicable
Trigger value of MACD indicator.	Property:trigger\$("#container").ejChart({ indicators: [{ trigger: 14 }] });	Not applicable
Fill color for indicator	Property:fill\$("#container").ejChart({ indicators: [{ fill: '#EEDDCC' }] });	Property:animation.enablelet chart: Chart = new Chart({ indicators: [{ fill: 'red' }] });
Width for indicator	Property:width\$("#container").ejChart({ indicators: [{ width: 2 }] });	Property:widthlet chart: Chart = new Chart({

		<code>indicators: [{ width: 3 }]]);</code>
xAxis Name of indicator	<code>Property:xAxisName\$("#container").ejChart({ indicators: [{ xAxisName: ' ' }]]);</code>	<code>Property:xAxisNamelet chart: Chart = new Chart({ indicators: [{ xAxisName: ' ' }]]);</code>
yAxis Name of indicator	<code>Property:yAxisName\$("#container").ejChart({ indicators: [{ yAxisName: ' ' }]]);</code>	<code>Property:yAxisNamelet chart: Chart = new Chart({ indicators: [{ yAxisName: ' ' }]]);</code>
Visibility of indicator	<code>Property:visibility\$("#container").ejChart({ indicators: [{ visibility: true }]]);</code>	Not applicable

Legend

<!-- markdownlint-disable MD033 -->

Behaviour	API in Essential JS 1	API in Essential JS 2
Default legend	<code>Property:visible\$("#container").ejChart({ legend: { visible: true } });</code>	<code>Property:visiblelet chart: Chart = new Chart({ legendSettings: { visible: true } });</code>
Legend height	<code>Property:size.height\$("#container").ejChart({ legend: { size : { height: 50 } } });</code>	<code>Property:heightlet chart: Chart = new Chart({ legendSettings: { height: '30' } });</code>
Legend width	<code>Property:size.width\$("#container").ejChart({ legend: { size: { width: 20 } } });</code>	<code>Property:widthlet chart: Chart = new Chart({ legendSettings: { width: '30' } });</code>
Legend location in chart	<code>Property:location\$("#container").ejChart({ legend: { location: { x: 3, y: 45 } } });</code>	<code>Property:heightlet chart: Chart = new Chart({ legendSettings: { location: { x: 3, y: 45 } } });</code>
Legend position in chart	<code>Property:position\$("#container").ejChart({ legend: { position: 'top' } });</code>	<code>Property:positionlet chart: Chart = new Chart({ legendSettings: { position: 'Top' } });</code>
Legend padding	Not applicable	<code>Property:paddinglet chart: Chart = new Chart({ legendSettings: { padding: 8 } });</code>
Legend alignment	<code>Property:position\$("#container").ejChart({ legend: { alignment: 'center' } });</code>	<code>Property:positionlet chart: Chart = new Chart({</code>

		<pre>legendSettings: { alignment: 'Center' }));</pre>
text style for legend	<pre>Property:font\$("#container").ejChart({ legend: { font: { fontFamily: '', fontWeight: '400', fontStyle: 'italic', size: '12px' } }});</pre>	<pre>Property:textStylelet chart: Chart = new Chart({ legendSettings: { textStyle: { size: '12px' , color: 'red', fontFamily: 'Italic', fontWeight: '400', fontStyle: 'Normal', opacity: 1, textAlignment: 'Center', textOverflow: 'Trim' } }});</pre>
shape height of legend	<pre>Property:itemStyle.height\$("#container").ejChart({ legend: { itemStyle: { height: 20 } } });</pre>	<pre>Property:shapeHeightlet chart: Chart = new Chart({ legendSettings: { shapeHeight: 20 } });</pre>
shape width of legend	<pre>Property:itemStyle.width\$("#container").ejChart({ legend: { itemStyle: { width: 20 } } });</pre>	<pre>Property:shapeWidthlet chart: Chart = new Chart({ legendSettings: { shapeWidth: 20 } });</pre>
shape width of legend	<pre>Property:itemStyle.width\$("#container").ejChart({ legend: { itemStyle: { width: 20 } } });</pre>	<pre>Property:shapeWidthlet chart: Chart = new Chart({ legendSettings: { shapeWidth: 20 } });</pre>
shape border of legend	<pre>Property:itemStyle.border\$("#container").ejChart({ legend: { itemStyle: { border: { width: 2, color: 'red' } } } });</pre>	Not Applicable
shape padding of legend	<pre>Property:itemPadding\$("#container").ejChart({ legend: { itemPadding: 10 } });</pre>	<pre>Property:shapePaddinglet chart: Chart = new Chart({ legendSettings: { shapePadding: 20 } });</pre>
Background of legend	<pre>Property:background\$("#container").ejChart({ legend: { background: 'transparent' } });</pre>	<pre>Property:backgorundlet chart: Chart = new Chart({ legendSettings: { background: 'transparent' }});</pre>
Opacity of legend	<pre>Property:opacity\$("#container").ejChart({ legend: { opacity: 0.3 } });</pre>	<pre>Property:opacitylet chart: Chart = new Chart({ legendSettings: { opacity: 0.4 } });</pre>
Toggle visibility of series while	<pre>Property:toggleSeriesVisibility\$("#container").ejCha rt({ legend: { toggleSeriesVisibility: true }});</pre>	<pre>Property:toggleVisibilitylet chart: Chart = new Chart({ legendSettings: { toggleVisibility: true }});</pre>

legend click		
Title for legend	Property:title\$("#container").ejChart({ legend: { title: { text: 'LegendTitle', font: { }, textAlign: 'middle' } } });	Not applicable
Text Overflow for legend	Property:title\$("#container").ejChart({ legend: { textOverflow: 'trim' } });	Property:textStyle.textOverflow let chart: new Chart({ legend: { text: { textOverflow: 'trim' } } });
Text width for legend while setting text overflow	Property:textWidth\$("#container").ejChart({ legend: { textWidth: 20 } });	Not applicable
Scroll bar for legend	Property:enableScrollBar\$("#container").ejChart({ legend: { enableScrollBar: true } });	Not applicable
Row count for legend	Property:rowCount\$("#container").ejChart({ legend: { rowCount: 2 } });	Not applicable
Column count for legend	Property:columnCount\$("#container").ejChart({ legend: { columnCount: 2 } });	Not applicable
Color for legend items	Property:fill\$("#container").ejChart({ legend: { fill: '#EEFFCC' } });	Not applicable

primaryXAxis

<!-- markdownlint-disable MD033 -->

Behaviour	API in Essential JS 1	API in Essential JS 2
Alternate grid band	Property:alternateGridBand\$("#container").ejChart({ primaryXAxis: { alternateGridBand: { even: { fill: 'red' } } } });	Not applicable
Axis line	Property:crossesAt\$("#container").ejChart({	Property:crossesAtlet chart: Chart = new Chart({ primaryXAxis: { crossesAt: 4 } });chart.appendTo('#chart');

cross value	primaryXAxis: { crossesAt: 0 }});	
axis name with which the axis line has to be crossed	Property:crossesInAxis\$("#container").ejChart({ primaryXAxis: { crossesInAxis: '' }});	Property:crossesInAxislet chart: Chart = new Chart({ primaryXAxis: { crossesInAxis: '' }});chart.appendTo('#chart');
axis elements placed with axis line	Property:showNextToAxisLine\$("#container").ejChart({ primaryXAxis: { showNextToAxisLine : true }});	Property:placeNextToAxisLinelet chart: Chart = new Chart({ primaryXAxis: { placeNextToAxisLine: '' }});chart.appendTo('#chart');
axis line style	Property:axisLine.color\$("#container").ejChart({ primaryXAxis: { axisLine: { color : 'red' }} }});	Property:lineStyle.colorlet chart: Chart = new Chart({ primaryXAxis: { lineStyle: { color: 'black' } }});chart.appendTo('#chart');
axis line dash Array	Property:axisLine.color\$("#container").ejChart({ primaryXAxis: { axisLine: { dashArray : '10, 5' }} }});	Property:lineStyle.dashArraylet chart: Chart = new Chart({ primaryXAxis: { lineStyle: { dashArray: '10, 5' } }});chart.appendTo('#chart');
Offset for axis	Property:axisLine.offset\$("#container").ejChart({ primaryXAxis: { axisLine: { offset : 10 }} }});	Property:plotOffsetlet chart: Chart = new Chart({ primaryXAxis: { plotOffset: 10 }});chart.appendTo('#chart');
Visible of	Property:axisLine.offset\$("#container").ejChart({	Property:visiblelet chart: Chart = new Chart({ primaryXAxis: { visible: false }});chart.appendTo('#chart');

an axis	primaryXAxis: { axisLine: { visible : false } }));	
Width of an axis	Property:axisLine.width width\$("#container").ejChart({ primaryXAxis: { axisLine: { width : 2 } } }));	Property:lineStyle.width let chart: Chart = new Chart({ primaryXAxis: { lineStyle: { width: 3 } } });chart.appendTo('#chart');
Column index of an axis	Property:columnIndex ex\$("#container").ejChart({ primaryXAxis: { columnIndex: 2 } }));	Property:columnIndex let chart: Chart = new Chart({ primaryXAxis: { columnIndex: 2 } });chart.appendTo('#chart');
span of an axis to place horizontally or vertically	Property:columnSpan an\$("#container").ejChart({ primaryXAxis: { columnIndex: 2 } }));	Property:span let chart: Chart = new Chart({ primaryXAxis: { span: 2 } });chart.appendTo('#chart');
Cross hair label of an axis	Property:crossHairLabel.visible \$("#container").ejChart({ primaryXAxis: { crossHairLabel: { visible: true } } }));	Property:crossHairTooltip.enable let chart: Chart = new Chart({ primaryXAxis: { crossHairTooltip: { enable: true } } });chart.appendTo('#chart');
Cross hair label color of an axis	Not applicable	Property:crossHairTooltip.fill let chart: Chart = new Chart({ primaryXAxis: { crossHairTooltip: { fill: 'red' } } });chart.appendTo('#chart');
Cross hair label text style	Not applicable	Property:crossHairTooltip.textStyle let chart: Chart = new Chart({ primaryXAxis: { crossHairTooltip: { textStyle: { } } } });chart.appendTo('#chart');

Desired interval count for primaryX Axis	Property:desiredIntervals\$("#container").ejChart({primaryXAxis: {desiredIntervals: 4}});	Property:desiredIntervalslet chart: Chart = new Chart({primaryXAxis: { desiredIntervals: 4}});chart.appendTo('#chart');
Edge primaryX Axis	Property:edgeLabelPlacement\$("#container").ejChart({primaryXAxis: {edgeLabelPlacement: 'none'}});	Property:edgeLabelPlacementlet chart: Chart = new Chart({primaryXAxis: { edgeLabelPlacement: 'Shift'}});chart.appendTo('#chart');
Enables trim for axis labels	Property:enableTrim\$("#container").ejChart({primaryXAxis: {enableTrim: true}});	Property:enableTrimlet chart: Chart = new Chart({primaryXAxis: { enableTrim: true}});chart.appendTo('#chart');
Specifies the interval of the axis according to the zoomed data of the chart	Property:enableAutoIntervalOnZooming\$("#container").ejChart({primaryXAxis: {enableAutoIntervalOnZooming: true}});	Property:enableAutoIntervalOnZoominglet chart: Chart = new Chart({ primaryXAxis: { enableAutoIntervalOnZooming: true}});chart.appendTo('#chart');
Specifies the interval of	Property:enableAutoIntervalOnZooming\$("#container").ejChart({primaryXAxis: {	Property:enableAutoIntervalOnZoominglet chart: Chart = new Chart({ primaryXAxis: { enableAutoIntervalOnZooming: true}});chart.appendTo('#chart');

the axis according to the zoomed data of the chart	<code>enableAutoIntervalOnZooming: true }));</code>	
Font style for primary X Axis	<code>Property:font\$("#container").ejChart({ primaryXAxis: { font: { fontFamily: 'Calibri', fontStyle: 'italic', fontWeight: '', opacity: 0.5, size: 12} } });</code>	<code>Property:titleStylelet chart: Chart = new Chart({ primaryXAxis: { titleStyle: { } } });chart.appendTo('#chart');</code>
Indexed for category axis	<code>Property:isIndexed\$("#container").ejChart({ primaryXAxis: { isIndexed: true } });</code>	<code>Property:isIndexedlet chart: Chart = new Chart({ primaryXAxis: { isIndexed: true } });chart.appendTo('#chart');</code>
Interval type for date time axis	<code>Property:intervalType\$("#container").ejChart({ primaryXAxis: { intervalType: 'Auto' } });</code>	<code>Property:intervalTypelet chart: Chart = new Chart({ primaryXAxis: { intervalType: 'Auto' } });chart.appendTo('#chart');</code>
Inversed axis	<code>Property:isInversed\$("#container").ejChart({ primaryXAxis: { isInversed: true } });</code>	<code>Property:isInversedlet chart: Chart = new Chart({ primaryXAxis: { isInversed: true } });chart.appendTo('#chart');</code>
Custom label	<code>Property:labelFormat\$("#container").ejChart({ primaryXAxis: {</code>	<code>Property:labelFormatlet chart: Chart = new Chart({ primaryXAxis: { labelFormat: '{value}K' } });chart.appendTo('#chart');</code>

format	labelFormat: '{value}K' }));	
label intersectAction	Property:labelIntersectAction\$("#container").ejChart({ primaryXAxis: { labelIntersectAction: 'trim' }});	Property:labelIntersectActionlet chart: Chart = new Chart({ primaryXAxis: { labelIntersectAction: 'Trim' }});chart.appendTo('#chart');
label Position	Property:labelPosition\$("#container").ejChart({ primaryXAxis: { labelPosition: 'inside' }));	Property:labelPositionlet chart: Chart = new Chart({ primaryXAxis: { labelPosition: 'Inside' }});chart.appendTo('#chart');
label Placement for category axis	Property:labelPlacement\$("#container").ejChart({ primaryXAxis: { labelPlacement: 'onTicks' }));	Property:labelPlacementlet chart: Chart = new Chart({ primaryXAxis: { labelPlacement: 'OnTicks' }});chart.appendTo('#chart');
Axis label alignment	Property:alignment\$("#container").ejChart({ primaryXAxis: { alignment: 'center' }));	Not Applicable
Rotation of axis labels	Property:labelRotation\$("#container").ejChart({ primaryXAxis: { labelRotation: 45 }));	Property:labelRotationlet chart: Chart = new Chart({ primaryXAxis: { labelRotation: 45 }});chart.appendTo('#chart');
Log base value for logarithmic axis	Property:logBase\$("#container").ejChart({ primaryXAxis: { logBase: 10 }});	Property:labelRotationlet chart: Chart = new Chart({ primaryXAxis: { logBase: 10 }));chart.appendTo('#chart');
Major grid line	Property:majorGridLines.visible\$("#container").ejChart({	Not Applicable

	<pre>primaryXAxis: { majorGridLines: { visible: true} }));</pre>	
Width of Major Grid Lines	<pre>Property:majorGridLines.width\$("#container").ejChart({ primaryXAxis: { majorGridLines: { width: 2} }});</pre>	<pre>Property:majorGridLines.widthlet chart: Chart = new Chart({ primaryXAxis: { majorGridLines: { width: 2} }});chart.appendTo('#chart');</pre>
Color of Major Grid Lines	<pre>Property:majorGridLines.color\$("#container").ejChart({ primaryXAxis: { majorGridLines: { color: 'black' } }));</pre>	<pre>Property:majorGridLines.colorlet chart: Chart = new Chart({ primaryXAxis: { majorGridLines: { color: 'black' } }});chart.appendTo('#chart');</pre>
Dash Array of Major Grid Lines	<pre>Property:majorGridLines.dashArray\$("#container").ejChart({ primaryXAxis: { majorGridLines: { dashArray: 'black' } }));</pre>	<pre>Property:majorGridLines.dashArraylet chart: Chart = new Chart({ primaryXAxis: { majorGridLines: { dashArray: 'black' } }});chart.appendTo('#chart');</pre>
Opacity of major grid line	<pre>Property:majorGridLines.opacity\$("#container").ejChart({ primaryXAxis: { majorGridLines: { opacity: true} }));</pre>	Not Applicable
Major Tick line	<pre>Property:majorTickLines.visible\$("#container").ejChart({ primaryXAxis: { majorTickLines: { visible: true} }));</pre>	Not Applicable
Width of Major Tick Lines	<pre>Property:majorTickLines.width\$("#container").ejChart({ primaryXAxis: { majorTickLines: { width: 2} }});</pre>	<pre>Property:majorTickLines.widthlet chart: Chart = new Chart({ primaryXAxis: { majorTickLines: { width: 2} }});chart.appendTo('#chart');</pre>

	<pre>{ width: 2} });</pre>	
Height of Major Tick Lines	<pre>Property:majorTickLines.size\$("#container").ejChart({ primaryXAxis: { majorTickLines: { size: 2 } } });</pre>	<pre>Property:majorTickLines.heightlet chart: Chart = new Chart({ primaryXAxis: { majorTickLines: { height: 2 } });chart.appendTo('#chart');</pre>
Color of Major Tick Lines	<pre>Property:majorTickLines.color\$("#container").ejChart({ primaryXAxis: { majorTickLines: { color: 'black' } } });</pre>	<pre>Property:majorTickLines.colorlet chart: Chart = new Chart({ primaryXAxis: { majorTickLines: { color: 'black' } });chart.appendTo('#chart');</pre>
Opacity of major Tick line	<pre>Property:majorTickLines.opacity\$("#container").ejChart({ primaryXAxis: { majorTickLines: { opacity: true } } });</pre>	Not Applicable
maximum labels of primary X Axis	<pre>Property:maximumLabels\$("#container").ejChart({ primaryXAxis: { maximumLabels: 5 } });</pre>	<pre>Property:maximumLabelslet chart: Chart = new Chart({ primaryXAxis: { maximumLabels: 4 });chart.appendTo('#chart');</pre>
maximum labels width of primary X Axis to trim	<pre>Property:maximumLabelWidth\$("#container").ejChart({ primaryXAxis: { maximumLabelWidth: 40 } });</pre>	<pre>Property:maximumLabelWidthlet chart: Chart = new Chart({ primaryXAxis: { maximumLabelWidth: 4 });chart.appendTo('#chart');</pre>
minor grid line	<pre>Property:minorGridLines.visible\$("#container").ejChart({</pre>	Not Applicable

	<pre>primaryXAxis: { minorGridLines: { visible: true} }));</pre>	
Width of minor Grid Lines	<pre>Property:minorGridLines.width\$("#container").ejChart({ primaryXAxis: { minorGridLines: { width: 2} } });</pre>	<pre>Property:minorGridLines.widthlet chart: Chart = new Chart({ primaryXAxis: { minorGridLines: { width: 2} });chart.appendTo('#chart');</pre>
Color of minor Grid Lines	<pre>Property:minorGridLines.color\$("#container").ejChart({ primaryXAxis: { minorGridLines: { color: 'black' } } });</pre>	<pre>Property:minorGridLines.colorlet chart: Chart = new Chart({ primaryXAxis: { minorGridLines: { color: 'black' } });chart.appendTo('#chart');</pre>
Dash Array of minor Grid Lines	<pre>Property:minorGridLines.dashArray\$("#container").ejChart({ primaryXAxis: { minorGridLines: { dashArray: 'black' } } });</pre>	<pre>Property:minorGridLines.dashArraylet chart: Chart = new Chart({ primaryXAxis: { minorGridLines: { dashArray: 'black' } });chart.appendTo('#chart');</pre>
Opacity of minor grid line	<pre>Property:minorGridLines.opacity\$("#container").ejChart({ primaryXAxis: { minorGridLines: { opacity: true} } });</pre>	Not Applicable
minor Tick line	<pre>Property:minorTickLines.visible\$("#container").ejChart({ primaryXAxis: { minorTickLines: { visible: true} } });</pre>	Not Applicable
Width of minor Tick Lines	<pre>Property:minorTickLines.width\$("#container").ejChart({ primaryXAxis: { minorTickLines: { width: 2} } });</pre>	<pre>Property:minorTickLines.widthlet chart: Chart = new Chart({ primaryXAxis: { minorTickLines: { width: 2} });chart.appendTo('#chart');</pre>

	<pre>{ width: 2} }});</pre>	
Height of minor Tick Lines	<pre>Property:minorTickLines.size\$("#container").ejChart({ primaryXAxis: { minorTickLines: { size: 2} }});</pre>	<pre>Property:minorTickLines.heightlet chart: Chart = new Chart({ primaryXAxis: { minorTickLines: { height: 2} }});chart.appendTo('#chart');</pre>
Color of minor Tick Lines	<pre>Property:minorTickLines.color\$("#container").ejChart({ primaryXAxis: { minorTickLines: { color: 'black' } }});</pre>	<pre>Property:minorTickLines.colorlet chart: Chart = new Chart({ primaryXAxis: { minorTickLines: { color: 'black' } }});chart.appendTo('#chart');</pre>
Opacity of minor Tick line	<pre>Property:minorTickLines.opacity\$("#container").ejChart({ primaryXAxis: { minorTickLines: { opacity: true} }});</pre>	Not Applicable
Minor ticks per interval of primary X Axis	<pre>Property:minorTicksPerInterval\$("#container").ejChart({ primaryXAxis: { minorTicksPerInterval: 4 }});</pre>	<pre>Property:minorTickLines.colorlet chart: Chart = new Chart({ primaryXAxis: { minorTicksPerInterval: 4 }});chart.appendTo('#chart');</pre>
name of the primary X Axis	<pre>Property:name\$("#container").ejChart({ primaryXAxis: { name: 'primaryXAxis' }});</pre>	<pre>Property:namelet chart: Chart = new Chart({ primaryXAxis: { name: 'primaryXAxis' }});chart.appendTo('#chart');</pre>
Orientation of primary X Axis	<pre>Property:orientation\$("#container").ejChart({ primaryXAxis: { orientation: 'Vertical' }});</pre>	Not Applicable

Plot offset for primaryX Axis	Property:plotOffset \$("#container").ejChart({ primaryXAxis: { plotOffset: 0 }});	Property:plotOffset let chart: Chart = new Chart({ primaryXAxis: { plotOffset: 0 }});chart.appendTo('#chart');
minimum for primaryX Axis	Property:range.minimum \$("#container").ejChart({ primaryXAxis: { range: { minimum: 10 }}});	Property:minimum let chart: Chart = new Chart({ primaryXAxis: { minimum: 23 }});chart.appendTo('#chart');
maximum for primaryX Axis	Property:range.maximum \$("#container").ejChart({ primaryXAxis: { range: { maximum: 10 }}});	Property:maximum let chart: Chart = new Chart({ primaryXAxis: { maximum: 23 }});chart.appendTo('#chart');
interval for primaryX Axis	Property:range.interval \$("#container").ejChart({ primaryXAxis: { range: { interval: 1 }}});	Property:interval let chart: Chart = new Chart({ primaryXAxis: { interval: 2 }});chart.appendTo('#chart');
RangePadding for primaryX Axis	Property:rangePadding \$("#container").ejChart({ primaryXAxis: { rangePadding: 'None' }});	Property:rangePadding let chart: Chart = new Chart({ primaryXAxis: { rangePadding: 'None' }});chart.appendTo('#chart');
Rounding Places in primaryX Axis	Property:roundingPlaces \$("#container").ejChart({ primaryXAxis: { roundingPlaces: 3 }});	Property:labelFormat let chart: Chart = new Chart({ primaryXAxis: { labelFormat: 'n3' }});chart.appendTo('#chart');
Scrollbar settings of	Property:scrollbarSettings \$("#container").ejChart({ primaryXAxis: {	Not Applicable

primaryX Axis	scrollbarSettings : { } }));	
TickPosition in primaryX Axis	Property:tickLinesPosition\$("#container").ejChart({ primaryXAxis: { tickLinesPosition: 'Inside' } }));	Property:tickPositionlet chart: Chart = new Chart({ primaryXAxis: { tickPosition: 'Inside' } });chart.appendTo('#chart');
valueType of primaryX Axis	Property:valueType\$("#container").ejChart({ primaryXAxis: { valueType: 'DateTime' } }));	Property:valueTypelet chart: Chart = new Chart({ primaryXAxis: { valueType: 'DateTime' } });chart.appendTo('#chart');
visible of primaryX Axis	Property:visible\$("#container").ejChart({ primaryXAxis: { visible: true } }));	Property:visiblelet chart: Chart = new Chart({ primaryXAxis: { visible: true } });chart.appendTo('#chart');
zoomFactor of primaryX Axis	Property:zoomFactor\$("#container").ejChart({ primaryXAxis: { zoomFactor: 0.3 } }));	Property:zoomFactorlet chart: Chart = new Chart({ primaryXAxis: { zoomFactor: 0.3 } });chart.appendTo('#chart');
zoomPosition of primaryX Axis	Property:zoomPosition\$("#container").ejChart({ primaryXAxis: { zoomPosition: 0.3 } }));	Property:zoomPositionlet chart: Chart = new Chart({ primaryXAxis: { zoomPosition: 0.3 } });chart.appendTo('#chart');
labelBorder of primaryX Axis	Property:labelBorder\$("#container").ejChart({ primaryXAxis: { labelBorder: { color: 'red', width: 2 } } }));	Property:borderlet chart: Chart = new Chart({ primaryXAxis: { border: { color: 'red', width: 3 } } });chart.appendTo('#chart');

title of primaryX Axis	Property:title.text\$ ("#container").ejChart({ primaryXAxis: { title: { text: 'Chart title' } }});	Property:titlelet chart: Chart = new Chart({ primaryXAxis: { title: 'Chart title' }});chart.appendTo('#chart');												
Strip Line of primaryX Axis	Property:stripLine\$ ("#container").ejChart({ primaryXAxis: { stripLine: [] }});	Property:stripLineslet chart: Chart = new Chart({ primaryXAxis: { stripLines: [] }});chart.appendTo('#chart');												
Multi level labels of primaryX Axis	Property:multiLevelLabels\$ ("#container").ejChart({ primaryXAxis: { multiLevelLabels: [] }});	Property:multiLevelLabelslet chart: Chart = new Chart({ primaryXAxis: { stripLines: [] }});chart.appendTo('#chart');												
skeleton for an axes	Not Applicable	Property:skeletonlet chart: Chart = new Chart({ axes: [{ skeleton: 'yMd' }]});chart.appendTo('#chart');												
skeleton type for an axes	Not Applicable	<div>Property:skeletonTypelet chart: Chart = new Chart({ axes: [{ skeletonType: 'DateTime' }]});chart.appendTo('#chart');</div> <div> <div>## primaryYAxis</div> <table> <tr> <td>Behaviour</td><td>API in Essential JS 1</td><td>API in Essential JS 2</td></tr> <tr> <td>Alternate grid band</td><td>Property:alternateGridBand\$ ("#container").ejChart({ primaryYAxis: { alternateGridBand: { even: { fill: 'red' } } } });</td><td>Not applicable</td></tr> <tr> <td>Axis line cross value</td><td>Property:crossesAt\$ ("#container").ejChart({ primaryYAxis: { crossesAt: 0 } });</td><td>Property:crossesAtlet chart: Chart = new Chart({ primaryYAxis: { crossesAt: 4 }});chart.append To('#chart');</td></tr> <tr> <td>axis name with which the</td><td>Property:crossesInAxis\$ ("#container").ejChart({ primaryYAxis: { crossesInAxis: ' ' } });</td><td>Property:crossesInAxis1 et chart: Chart = new Chart({ primaryYAxis: {</td></tr> </table> </div>	Behaviour	API in Essential JS 1	API in Essential JS 2	Alternate grid band	Property:alternateGridBand\$ ("#container").ejChart({ primaryYAxis: { alternateGridBand: { even: { fill: 'red' } } } });	Not applicable	Axis line cross value	Property:crossesAt\$ ("#container").ejChart({ primaryYAxis: { crossesAt: 0 } });	Property:crossesAtlet chart: Chart = new Chart({ primaryYAxis: { crossesAt: 4 }});chart.append To('#chart');	axis name with which the	Property:crossesInAxis\$ ("#container").ejChart({ primaryYAxis: { crossesInAxis: ' ' } });	Property:crossesInAxis1 et chart: Chart = new Chart({ primaryYAxis: {
Behaviour	API in Essential JS 1	API in Essential JS 2												
Alternate grid band	Property:alternateGridBand\$ ("#container").ejChart({ primaryYAxis: { alternateGridBand: { even: { fill: 'red' } } } });	Not applicable												
Axis line cross value	Property:crossesAt\$ ("#container").ejChart({ primaryYAxis: { crossesAt: 0 } });	Property:crossesAtlet chart: Chart = new Chart({ primaryYAxis: { crossesAt: 4 }});chart.append To('#chart');												
axis name with which the	Property:crossesInAxis\$ ("#container").ejChart({ primaryYAxis: { crossesInAxis: ' ' } });	Property:crossesInAxis1 et chart: Chart = new Chart({ primaryYAxis: {												

		<p>axis line has to be crossed</p> <p>axis elements placed with axis line</p> <p>axis line style</p> <p>axis line dashArray</p> <p>Offset for axis</p> <p>Visible of an axis</p> <p>Width of an axis</p>	<pre> crossesInAxis: '' });chart.appendT o('#chart'); Property:placeNextToA xisLinelet chart: Chart = new Chart({ primaryYAxis: { placeNextToAxisLi ne: '' });chart.appendT o('#chart'); Property:lineStyle.color let chart: Chart = new Chart({ primaryYAxis: { lineStyle: { color: 'black' } });chart.appendT o('#chart'); Property:lineStyle.dash Arraylet chart: Chart = new Chart({ primaryYAxis: { lineStyle: { dashArray: '10, 5' } });chart.appendT o('#chart'); Property:plotOffsetlet chart: Chart = new Chart({ primaryYAxis: { plotOffset: 10 });chart.appendT o('#chart'); Property:visiblelet chart: Chart = new Chart({ primaryYAxis: { visible: false });chart.appendT o('#chart'); Property:lineStyle.widt hlet chart: Chart = new Chart({ primaryYAxis: { lineStyle: { </pre>
--	--	--	---

		<pre>width: 3 } });chart.appendT o('#chart');</pre>
Column index of an axis	Property:columnIndex\$("#container").ejChart({ primaryYAxis: { columnIndex: 2 }});	<pre>Property:columnIndex1 et chart: Chart = new Chart({ primaryYAxis: { columnIndex: 2 });chart.appendT o('#chart');</pre>
span of an axis to place horizontally or vertically	Property:columnSpan\$("#container").ejChart({ primaryYAxis: { columnIndex: 2 }});	<pre>Property:spanlet chart: Chart = new Chart({ primaryYAxis: { span: 2 });chart.appendT o('#chart');</pre>
Crosshair label of an axis	Property:crossHairLabel.visible\$("#container").ejChart({ primaryYAxis: { crossHairLabel: { visible: true }}});	<pre>Property:crossHairToolt ip.enablelet chart: Chart = new Chart({ primaryYAxis: { crossHairTooltip: { enable: true }}});chart.append To('#chart');</pre>
Crosshair label color of an axis	Not applicable	<pre>Property:crossHairToolt ip.filllet chart: Chart = new Chart({ primaryYAxis: { crossHairTooltip: { fill: 'red' }}});chart.append To('#chart');</pre>
Crosshair label text style	Not applicable	<pre>Property:crossHairToolt ip.textStylelet chart: Chart = new Chart({ primaryYAxis: { crossHairTooltip: { textStyle: { } }}});chart.append To('#chart');</pre>
Desired interval count for	Property:desiredIntervals\$("#container").ejChart({ primaryYAxis: { desiredIntervals: 4 }});	<pre>Property:desiredInterva Islet chart: Chart = new Chart({ primaryYAxis: { desiredIntervals:</pre>

		<pre> primaryYAxis 4 });chart.appendT o('#chart'); Property:edgeLabelPlac ementlet chart: Edges Property:edgeLabelPlacement\$("#con Chart = new primaryYAxis primaryYAxis: { Chart({ axis edgeLabelPlacement: 'none' primaryYAxis: { }); edgeLabelPlacemen }); t: 'Shift' });chart.appendT o('#chart'); Property:enableTrimle t chart: Chart = Enables Property:enableTrim\$("#container" trim for).ejChart({ primaryYAxis: { axis labels enableTrim: true }); Specifies Property:enableAutoIntervalOnZoomin the g\$("#container").ejChart({ interval of primaryYAxis: { the axis enableAutoIntervalOnZooming: according true }); to the zoomed data of the chart Specifies Property:enableAutoIntervalOnZoomin the g\$("#container").ejChart({ interval of primaryYAxis: { the axis enableAutoIntervalOnZooming: according true }); to the zoomed data of the chart Font style Property:font\$("#container").ejC for hart({ primaryYAxis: { font: primaryYAxis { fontFamily: 'Calibri', axis fontStyle: 'italic', fontWeight: '', opacity: 0.5, size: 12} }); Property:titleStylelet chart: Chart = new Chart({ primaryYAxis: { titleStyle: { } });chart.appendT o('#chart'); </pre>
--	--	---

		<p>Indexed for category axis</p> <p>Property:isIndexed\$("#container").ejChart({ primaryYAxis: { isIndexed: true }});</p> <p>Interval type for date time axis</p> <p>Property:intervalType\$("#container").ejChart({ primaryYAxis: { intervalType: 'Auto' }});</p> <p>Inversed axis</p> <p>Property:isInversed\$("#container").ejChart({ primaryYAxis: { isInversed: true }});</p> <p>Custom label format</p> <p>Property:labelFormat\$("#container").ejChart({ primaryYAxis: { labelFormat: '{value}K' }});</p> <p>labelIntersectAction</p> <p>Property:labelIntersectAction\$("#container").ejChart({ primaryYAxis: { labelIntersectAction: 'trim' }});</p> <p>labelPosition</p> <p>Property:labelPosition\$("#container").ejChart({ primaryYAxis: { labelPosition: 'inside' }});</p> <p>labelPlacement for</p> <p>Property:labelPlacement\$("#container").ejChart({ primaryYAxis: {</p>	<p>Property:isIndexedlet chart: Chart = new Chart({ primaryYAxis: { isIndexed: true }});chart.appendTo('#chart');</p> <p>Property:intervalTypelet chart: Chart = new Chart({ primaryYAxis: { intervalType: 'Auto' }});chart.appendTo('#chart');</p> <p>Property:isInversedlet chart: Chart = new Chart({ primaryYAxis: { isInversed: true }});chart.appendTo('#chart');</p> <p>Property:labelFormatlet chart: Chart = new Chart({ primaryYAxis: { labelFormat: '{value}K' }});chart.appendTo('#chart');</p> <p>Property:labelIntersectActionlet chart: Chart = new Chart({ primaryYAxis: { labelIntersectAction: 'Trim' }});chart.appendTo('#chart');</p> <p>Property:labelPositionlet chart: Chart = new Chart({ primaryYAxis: { labelPosition: 'Inside' }});chart.appendTo('#chart');</p> <p>Property:labelPlacementlet chart: Chart =</p>
--	--	--	---

		<pre>category axis { labelPlacement: 'onTicks' })); = new Chart({ primaryYAxis: { labelPlacement: 'OnTicks' }});chart.appendT o('#chart');</pre>
	Axis label alignment	<pre>Property:alignment\$("#container") .ejChart({ primaryYAxis: { alignment: 'center' }});</pre> <p>Not Applicable</p>
	Rotation of axis labels	<pre>Property:labelRotation\$("#containe r").ejChart({ primaryYAxis: { labelRotation: 45 }});</pre> <p>Property:labelRotation let chart: Chart = new Chart({ primaryYAxis: { labelRotation: 45 }});chart.appendT o('#chart');</p>
	Log base value for logarithmic axis	<pre>Property:logBase\$("#container").e jChart({ primaryYAxis: { logBase: 10 }});</pre> <p>Property:labelRotation let chart: Chart = new Chart({ primaryYAxis: { logBase: 10 }});chart.appendT o('#chart');</p>
	Major grid line	<pre>Property:majorGridLines.visible\$("#co ntainer").ejChart({ primaryYAxis: { majorGridLines: { visible: true} }});</pre> <p>Not Applicable</p>
	Width of MajorGrid Lines	<pre>Property:majorGridLines.width\$("#co ntainer").ejChart({ primaryYAxis: { majorGridLines: { width: 2} }});</pre> <p>Property:majorGridLine s.widthlet chart: Chart = new Chart({ primaryYAxis: { majorGridLines: { width: 2 }});chart.appendT o('#chart');</p>
	Color of MajorGrid Lines	<pre>Property:majorGridLines.color\$("#con tainer").ejChart({ primaryYAxis: { majorGridLines: { color: 'black' } }});</pre> <p>Property:majorGridLine s.colorlet chart: Chart = new Chart({ primaryYAxis: { majorGridLines: { color: 'black' } }});chart.appendT o('#chart');</p>
	DashArray of	<pre>Property:majorGridLines.dashArray\$("# #container").ejChart({ primaryYAxis: {</pre> <p>Property:majorGridLine s.dashArraylet</p>

		<p>MajorGrid Lines</p> <pre>majorGridLines: { dashArray: 'black' } }));</pre> <p>Opacity of major grid line</p> <pre>Property:majorGridLines.opacity\$("#container").ejChart({ primaryYAxis: { majorGridLines: { opacity: true} } }));</pre> <p>Major Tick line</p> <pre>Property:majorTickLines.visible\$("#container").ejChart({ primaryYAxis: { majorTickLines: { visible: true} } }));</pre> <p>Width of MajorTick Lines</p> <pre>Property:majorTickLines.width\$("#container").ejChart({ primaryYAxis: { majorTickLines: { width: 2} } }));</pre> <p>Height of MajorTick Lines</p> <pre>Property:majorTickLines.size\$("#container").ejChart({ primaryYAxis: { majorTickLines: { size: 2} } }));</pre> <p>Color of MajorTick Lines</p> <pre>Property:majorTickLines.color\$("#container").ejChart({ primaryYAxis: { majorTickLines: { color: 'black' } } }));</pre> <p>Opacity of major Tick line</p> <pre>Property:majorTickLines.opacity\$("#container").ejChart({ primaryYAxis: {</pre>	<pre>chart: Chart = new Chart({ primaryYAxis: { majorGridLines: { dashArray: 'black' } } });chart.appendTo('#chart');</pre> <p>Not Applicable</p> <p>Not Applicable</p> <pre>Property:majorTickLines.widthlet chart: Chart = new Chart({ primaryYAxis: { majorTickLines: { width: 2} } });chart.appendTo('#chart');</pre> <pre>Property:majorTickLines.heightlet chart: Chart = new Chart({ primaryYAxis: { majorTickLines: { height: 2} } });chart.appendTo('#chart');</pre> <pre>Property:majorTickLines.colorlet chart: Chart = new Chart({ primaryYAxis: { majorTickLines: { color: 'black' } } });chart.appendTo('#chart');</pre> <p>Not Applicable</p>
--	--	--	---

		<pre>majorTickLines: { opacity: true} }));</pre>	
	maximum labels of primaryYAxis	<pre>Property:maximumLabels\$("#container").ejChart({ primaryYAxis: { maximumLabels: 5 } });</pre>	<pre>Property:maximumLabelslet chart: Chart = new Chart({ primaryYAxis: { maximumLabels: 4 }});chart.appendTo('#chart');</pre>
	maximum labels width of primaryYAxis to trim	<pre>Property:maximumLabelWidth\$("#container").ejChart({ primaryYAxis: { maximumLabelWidth: 40 } });</pre>	<pre>Property:maximumLabelWidthlet chart: Chart = new Chart({ primaryYAxis: { maximumLabelWidth : 4 }});chart.appendTo('#chart');</pre>
	minor grid line	<pre>Property:minorGridLines.visible\$("#container").ejChart({ primaryYAxis: { minorGridLines: { visible: true} }));</pre>	Not Applicable
	Width of minorGrid Lines	<pre>Property:minorGridLines.width\$("#container").ejChart({ primaryYAxis: { minorGridLines: { width: 2} }});</pre>	<pre>Property:minorGridLines.widthlet chart: Chart = new Chart({ primaryYAxis: { minorGridLines: { width: 2} }});chart.appendTo('#chart');</pre>
	Color of minorGrid Lines	<pre>Property:minorGridLines.color\$("#container").ejChart({ primaryYAxis: { minorGridLines: { color: 'black' } } });</pre>	<pre>Property:minorGridLines.colorlet chart: Chart = new Chart({ primaryYAxis: { minorGridLines: { color: 'black' } }});chart.appendTo('#chart');</pre>
	DashArray of minorGrid Lines	<pre>Property:minorGridLines.dashArray\$("#container").ejChart({ primaryYAxis: { minorGridLines: { dashArray: 'black' } } });</pre>	<pre>Property:minorGridLines.dashArraylet chart: Chart = new Chart({ primaryYAxis: { minorGridLines: { dashArray:</pre>

		<pre>'black' } });chart.appendT o('#chart');</pre>
Opacity of minor grid line	Property:minorGridLines.opacity\$ ("#c ontainer").ejChart({ primaryYAxis: { minorGridLines: { opacity: true} }));	Not Applicable
minor Tick line	Property:minorTickLines.visible\$ ("#co ntainer").ejChart({ primaryYAxis: { minorTickLines: { visible: true} }));	Not Applicable
Width of minor Tick Lines	Property:minorTickLines.width\$ ("#co ntainer").ejChart({ primaryYAxis: { minorTickLines: { width: 2} }));	Property:minorTickLine s.width let chart: Chart = new Chart({ primaryYAxis: { minorTickLines: { width: 2} });chart.appendT o('#chart');
Height of minor Tick Lines	Property:minorTickLines.size\$ ("#cont ainer").ejChart({ primaryYAxis: { minorTickLines: { size: 2} }));	Property:minorTickLine s.height let chart: Chart = new Chart({ primaryYAxis: { minorTickLines: { height: 2} });chart.appendT o('#chart');
Color of minor Tick Lines	Property:minorTickLines.color\$ ("#con tainer").ejChart({ primaryYAxis: { minorTickLines: { color: 'black' } }));	Property:minorTickLine s.color let chart: Chart = new Chart({ primaryYAxis: { minorTickLines: { color: 'black' } });chart.appendT o('#chart');
Opacity of minor Tick line	Property:minorTickLines.opacity\$ ("#c ontainer").ejChart({ primaryYAxis: { minorTickLines: { opacity: true} }));	Not Applicable
Minor ticks per interval of	Property:minorTicksPerInterval\$ ("#co ntainer").ejChart({	Property:minorTickLine s.color let chart: Chart = new

		<pre> primaryYAxis: { minorTicksPerInterval: 4 }}); </pre>	<pre> Chart({ primaryYAxis: { minorTicksPerInterval: 4 }}); chart.appendT o('#chart'); </pre>
	<p>name of the primaryYAxis</p>	<pre> Property:name\$("#container").ej Chart({ primaryYAxis: { name: 'primaryYAxis' }}); </pre>	<pre> Property:namelet chart: Chart = new Chart({ primaryYAxis: { name: 'primaryYAxis' }}); chart.appendT o('#chart'); </pre>
	<p>Orientation of primaryYAxis</p>	<pre> Property:orientation\$("#container").ejChart({ primaryYAxis: { orientation: 'Vertical' }}); </pre>	Not Applicable
	<p>Plot offset for primaryYAxis</p>	<pre> Property:plotOffset\$("#container") .ejChart({ primaryYAxis: { plotOffset: 0 }}); </pre>	<pre> Property:plotOffsetlet chart: Chart = new Chart({ primaryYAxis: { plotOffset: 0 }}); chart.appendT o('#chart'); </pre>
	<p>minimum for primaryYAxis</p>	<pre> Property:range.minimum\$("#contai ner").ejChart({ primaryYAxis: { range: { minimum: 10 }}}); </pre>	<pre> Property:minimumlet chart: Chart = new Chart({ primaryYAxis: { minimum: 23 }}); chart.appendT o('#chart'); </pre>
	<p>maximum for primaryYAxis</p>	<pre> Property:range.maximum\$("#contai ner").ejChart({ primaryYAxis: { range: { maximum: 10 }}}); </pre>	<pre> Property:maximumlet chart: Chart = new Chart({ primaryYAxis: { maximum: 23 }}); chart.appendT o('#chart'); </pre>
	<p>interval for primaryYAxis</p>	<pre> Property:range.interval\$("#containe r").ejChart({ primaryYAxis: { range: { interval: 1 }}}); </pre>	<pre> Property:intervallet chart: Chart = new Chart({ primaryYAxis: { interval: 2 }}); chart.appendT o('#chart'); </pre>

		<p>RangePadding for primaryYAxis</p> <p>Property:rangePadding <pre>let chart: Chart = new Chart({ primaryYAxis: { rangePadding: 'None' } }); chart.appendTo('#chart');</pre> </p> <p>Rounding Places in primaryYAxis</p> <p>Property:roundingPlaces <pre>\$("#container").ejChart({ primaryYAxis: { roundingPlaces: 3 } });</pre> </p> <p>ScrollBar settings of primaryYAxis</p> <p>Property:scrollbarSettings <pre>\$("#container").ejChart({ primaryYAxis: { scrollbarSettings : { } } });</pre> </p> <p>TickPosition in primaryYAxis</p> <p>Property:tickLinesPosition <pre>\$("#container").ejChart({ primaryYAxis: { tickLinesPosition: 'Inside' } });</pre> </p> <p>valueType of primaryYAxis</p> <p>Property:valueType <pre>\$("#container").ejChart({ primaryYAxis: { valueType: 'DateTime' } });</pre> </p> <p>visible of primaryYAxis</p> <p>Property:visible <pre>\$("#container").ejChart({ primaryYAxis: { visible: true } });</pre> </p> <p>zoomFactor of primaryYAxis</p> <p>Property:zoomFactor <pre>\$("#container").ejChart({ primaryYAxis: { zoomFactor: 0.3 } });</pre> </p>	<p>Property:rangePadding <pre>let chart: Chart = new Chart({ primaryYAxis: { rangePadding: 'None' } }); chart.appendTo('#chart');</pre> </p> <p>Property:labelFormat <pre>let chart: Chart = new Chart({ primaryYAxis: { labelFormat: 'n3' } }); chart.appendTo('#chart');</pre> </p> <p>Not Applicable</p> <p>Property:tickPosition <pre>let chart: Chart = new Chart({ primaryYAxis: { tickPosition: 'Inside' } }); chart.appendTo('#chart');</pre> </p> <p>Property:valueType <pre>let chart: Chart = new Chart({ primaryYAxis: { valueType: 'DateTime' } }); chart.appendTo('#chart');</pre> </p> <p>Property:visible <pre>let chart: Chart = new Chart({ primaryYAxis: { visible: true } }); chart.appendTo('#chart');</pre> </p> <p>Property:zoomFactor <pre>let chart: Chart = new Chart({ primaryYAxis: { zoomFactor: 0.3 } }); chart.appendTo('#chart');</pre> </p>
--	--	---	---

		<p>zoomPosition of primaryYAxis</p> <p>Property:zoomPosition <pre>Property:zoomPosition\$("#container" r").ejChart({ primaryYAxis: { zoomPosition: 0.3 }});</pre></p> <p>Property:zoomPosition <pre>let chart: Chart = new Chart({ primaryYAxis: { zoomPosition: 0.3 }});chart.appendT o('#chart');</pre></p> <p>labelBorder of primaryYAxis</p> <p>Property:labelBorder <pre>Property:labelBorder\$("#container").ejChart({ primaryYAxis: { labelBorder: { color: 'red', width: 2 } }});</pre></p> <p>Property:labelBorder <pre>let chart: Chart = new Chart({ primaryYAxis: { border: { color: 'red', width: 3 } }});chart.appendT o('#chart');</pre></p> <p>title of primaryYAxis</p> <p>Property:title <pre>Property:title.text\$("#container"). ejChart({ primaryYAxis: { title: { text: 'Chart title' } }});</pre></p> <p>Property:title <pre>let chart: Chart = new Chart({ primaryYAxis: { title: 'Chart title' }});chart.appendT o('#chart');</pre></p> <p>StripLine of primaryYAxis</p> <p>Property:stripLine <pre>Property:stripLine\$("#container"). ejChart({ primaryYAxis: { stripLine: [] }});</pre></p> <p>Property:stripLines <pre>let chart: Chart = new Chart({ primaryYAxis: { stripLines: [] }});chart.appendT o('#chart');</pre></p> <p>Multilevel labels of primaryYAxis</p> <p>Property:multiLevelLabels <pre>Property:multiLevelLabels\$("#contai ner").ejChart({ primaryYAxis: { multiLevelLabels: [] }});</pre></p> <p>Property:multiLevelLabels <pre>let chart: Chart = new Chart({ primaryYAxis: { stripLines: [] }});chart.appendT o('#chart');</pre></p> <p>skeleton for an axes</p> <p>Not Applicable</p> <p>Property:skeleton <pre>let chart: Chart = new Chart({ axes: [{ skeleton: 'yMd' }]});chart.append To('#chart');</pre></p> <p>skeleton type for an axes</p> <p>Not Applicable</p> <p>Property:skeletonType <pre>let chart: Chart = new Chart({ axes: [{ skeletonType:</pre></p>
--	--	---

		'DateTime' }}});chart.append To('#chart');
	## Axes	
Beha viour	API in Essential JS 1 API in Essential JS 2	
	Property:alternateGridBand	
Alter nate grid band	\$("#container") .ejChart({ axes: [{ Not applicable alternateGridBand: { even: { fill: 'red' } }] });	
Axis line cross value	Property:crossesAt \$("#container") .ejChart({ axes: [{ crossesAt: 0 }] });	Property:crossesAtlet chart: Chart = new Chart({ axes: [{ crossesAt: 4}] });chart.appendTo('#chart');
axis nam e with whic h the axis line has to be cross ed	Property:crossesInAxis \$("#container").ejChart(axes: [{ crossesInAxis: '' }] });	Property:crossesInAxislet chart: Chart = new Chart({ axes: [{ crossesInAxis: '' }] });chart.appendTo('#chart');
axis elem ents place d with axis line	Property:showNextToAxisLine \$("#container") .ejChart({ axes: [{ showNextToAxisLine : true }] });	Property:placeNextToAxisLinelet chart: Chart = new Chart({ axes: [{ placeNextToAxisLine: '' }] });chart.appendTo('#chart');
axis line style	Property:axisLineColor \$("#container") .ejChart({ axes: [{	Property:lineStyle.colorlet chart: Chart = new Chart({ axes: [{ lineStyle: { color: 'black' } }] });chart.appendTo('#chart');

		<pre> axisLine: { color : 'red' } }}}); Property:axisLine.c olor\$("#containe r").ejChart({ axes: [{ axisLine: { dashArray : '10, 5' } }]}); Property:axisLine.of fset\$("#containe r").ejChart({ axes: [{ axisLine: { offset : 10 } }]}); Property:axisLine.of fset\$("#containe r").ejChart({ axes: [{ axisLine: { visible : false } }]}); Property:axisLine.w idth\$("#containe r").ejChart({ axes: [{ axisLine: { width : 2 } }]}); Property:columnInd ex\$("#container ").ejChart({ axes: [{ columnIndex: 2 }]}); Property:columnSp an\$("#container ").ejChart({ axes: [{ columnIndex: 2 }]}); </pre>
		<pre> Property:lineStyle.dashArraylet chart: Chart = new Chart({ axes: [{ lineStyle: { dashArray: '10, 5' } }]});chart.appendTo('#chart'); Property:plotOffsetlet chart: Chart = new Chart({ axes: [{ plotOffset: 10 }]});chart.appendTo('#chart'); Property:visiblelet chart: Chart = new Chart({ axes: [{ visible: false }]});chart.appendTo('#chart'); Property:lineStyle.widthlet chart: Chart = new Chart({ axes: [{ lineStyle: { width: 3 } }]});chart.appendTo('#chart'); Property:columnIndexlet chart: Chart = new Chart({ axes: [{ columnIndex: 2 }]});chart.appendTo('#chart'); Property:spanlet chart: Chart = new Chart({ axes: [{ span: 2 }]});chart.appendTo('#chart'); </pre>

		<p>Property:crossHairLabel visible\$("#container").ejChart({ axes: [{ crossHairLabel: { visible: true } }] });</p> <p>Property:crossHairTooltip.enable let chart: Chart = new Chart({ axes: [{ crossHairTooltip: { enable: true } }] });chart.appendTo('#chart');</p>
		<p>Property:crossHairTooltip.fill let chart: Chart = new Chart({ axes: [{ crossHairTooltip: { fill: 'red' } }] });chart.appendTo('#chart');</p>
		<p>Property:crossHairTooltip.textStyle let chart: Chart = new Chart({ axes: [{ crossHairTooltip: { textStyle: { } } }] });chart.appendTo('#chart');</p>
		<p>Property:desiredIntervals let chart: Chart = new Chart({ axes: [{ desiredIntervals: 4 } }] });chart.appendTo('#chart');</p>
		<p>Property:edgeLabelPlacement let chart: Chart = new Chart({ axes: [{ edgeLabelPlacement: 'Shift' } }] });chart.appendTo('#chart');</p>
		<p>Property:enableTrim let chart: Chart = new Chart({ axes: [{ enableTrim: true } }] });chart.appendTo('#chart');</p>
		<p>Property:enableAutoIntervalOnZooming let chart: Chart = new Chart({ axes: [{ enableAutoIntervalOnZooming: true } }] });chart.appendTo('#chart');</p>

		<pre> val of axes: [{ the enableAutoInter axis valOnZooming: true }]]); accor ding to the zoo med data of the chart Speci fies the inter val of the Property:enableAut axis olIntervalOnZoomin accor g\$("#container" Property:enableAutoIntervalOnZoominglet ding).ejChart({ chart: Chart = new Chart({ axes: [{ to axes: [{ enableAutoIntervalOnZooming: true the enableAutoInter }]]);chart.appendTo('#chart'); zoo valOnZooming: med true }]]); data of the chart Property:font\$("## container").ejC Font hart({ axes: [style { font: { for fontFamily: Property:titleStylelet chart: Chart = new prim 'Calibri', Chart({ axes: [{ titleStyle: { } aryY fontStyle: }]]);chart.appendTo('#chart'); Axis 'italic', fontWeight: '', opacity: 0.5, size: 12} }]]); Property:isIndexed Index \$\$("#container") Property:isIndexedlet chart: Chart = new ed .ejChart({ Chart({ axes: [{ isIndexed: true for axes: [{ categ isIndexed: true }]]);chart.appendTo('#chart'); }]]); </pre>
--	--	---

		<pre> ory axis Inter val Property:intervalType type pe\$("#container for ").ejChart({ date axes: [{ time intervalType}: axis 'Auto' }]}); Property:isInversed Inver \$("#container") sed .ejChart({ axis axes: [{ Property:isInversedlet chart: Chart = new Chart({ axes: [{ isInversed: true isInversed: }]);chart.appendTo('#chart'); true }]}); Property:labelForm Cust at\$("#container om ").ejChart({ label axes: [{ form labelFormat: at '{value}K' }]}); Property:labelInter label ectAction\$("#con nters tainer").ejChar ectA t({ axes: [{ ction labelIntersectA ction: 'trim' }]}); Property:labelPositi label on\$("#container Positi ").ejChart({ on axes: [{ Property:labelPositionlet chart: Chart = new Chart({ axes: [{ labelPosition: labelPosition: 'Inside' 'inside' }]});chart.appendTo('#chart'); 'inside' }]}); label Place Property:labelPlace ment ment\$("#contain for er").ejChart({ categ axes: [{ ory labelPlacement: axis 'onTicks' }]}); Property:alignment Axis \$("#container") label .ejChart({ axes: [{ </pre>
--	--	--

		<pre> align alignment: ment 'center']]]); Rotation\$("#container").ejChart({ axes: [{ labelRotation: 45 }]]]); Property:labelRotation Property:labelRotation let chart: Chart = new Chart({ axes: [{ labelRotation: 45 }]]]);chart.appendTo('#chart'); Log base Property:logBase\$ (value "#container").ejChart({ axes: [{ logBase: 10 }]]]);chart.appendTo('#chart'); for jChart({ axes: new Chart({ axes: [{ logBase: 10 logar [{ logBase: 10 }]]]);chart.appendTo('#chart'); ithmi }]]]); c axis Property:majorGrid Lines.visible\$("#container").ejChart({ axes: [{ majorGridLines: { visible: true } }]]]); Not Applicable Property:majorGrid Lines.width\$("#container").ejChart({ axes: [{ majorGridLines: { width: 2 } }]]]); Property:majorGridLines.width let chart: Chart = new Chart({ axes: [{ majorGridLines: { width: 2 } }]]]);chart.appendTo('#chart'); Property:majorGrid Lines.color\$("#container").ejChart({ axes: [{ majorGridLines: { color: 'black' } }]]]); Property:majorGridLines.color let chart: Chart = new Chart({ axes: [{ majorGridLines: { color: 'black' } }]]]);chart.appendTo('#chart'); Property:majorGrid Lines.dashArray\$("#container").ejChart({ axes: [{ majorGridLines: { dashArray: 'black' } }]]]); Property:majorGridLines.dashArray let chart: Chart = new Chart({ axes: [{ majorGridLines: { dashArray: 'black' } }]]]);chart.appendTo('#chart'); Property:majorGrid Lines.opacity\$("#container").ejChart({ axes: [{ majorGridLines: { opacity: 0.5 } }]]]); Not Applicable </pre>
--	--	--

		<pre> majorContainer").ejChart({ axes: [{ majorGridLines: line { opacity: true}]}); Property:majorTick Lines.visible\$("#co MajorContainer").ejChart({ axes: [{ majorTickLines: line { visible: true}]}); Property:majorTick WidthLines.width\$("#co MajorContainer").ejChart({ axes: [{ majorTickLines: Lines { width: 2}] }); Property:majorTick HeightLines.size\$("#co MajorContainer").ejChart({ axes: [{ majorTickLines: Lines { size: 2}] }); Property:majorTick ColorLines.color\$("#co MajorContainer").ejChart({ axes: [{ majorTickLines: Lines { color: 'black' }] }); Property:majorTick OpacityLines.opacity\$("#co MajorContainer").ejChart({ axes: [{ majorTickLines: line { opacity: true}] }); maximumProperty:maximum labelLabels\$("#co MajorContainer").ejChart({ axes: [{ maximumLabels: 4 primmaximumLabels: aryY5 }] }); Axis </pre>
--	--	---

		<pre> maximum label Property:maximum LabelWidth\$("#co Property:maximumLabelWidthlet chart: ntainer").ejCha Chart = new Chart({ axes: [{ rt({ axes: [{ maximumLabelWidth: 4 maximumLabelWid }}}];chart.appendTo('#chart'); th: 40 }]]}); Axis to trim Property:minorGrid Lines.visible\$("#co ntainer").ejCha r grid rt({ axes: [{ Not Applicable line minorGridLines: { visible: true} }]]}); Property:minorGrid Width Lines.width\$("#co Property:minorGridLines.widthlet chart: h of ntainer").ejCha Chart = new Chart({ axes: [{ mino rt({ axes: [{ minorGridLines: { width: 2} rGrid minorGridLines: }]]];chart.appendTo('#chart'); Lines { width: 2} }]]}); Property:minorGrid Color Lines.color\$("#con Property:minorGridLines.colorlet chart: of tainer").ejChar Chart = new Chart({ axes: [{ mino t({ axes: [{ minorGridLines: { color: 'black' } rGrid minorGridLines: }]]];chart.appendTo('#chart'); Lines { color: 'black' } }]]}); Property:minorGrid Dash Lines.dashArray\$("# Array #container").ej Property:minorGridLines.dashArraylet chart: of Chart({ axes: [Chart = new Chart({ axes: [{ mino { minorGridLines: { dashArray: 'black' rGrid minorGridLines: } }]]];chart.appendTo('#chart'); Lines { dashArray: 'black' } }]]}); Property:minorGrid Opac Lines.opacity\$("#c ity of ontainer").ejCh mino art({ axes: [{ Not Applicable r grid minorGridLines: line { opacity: true} }]]}); </pre>
--	--	---

		<p>Property:minorTick Lines.visible\$("#container").ejChart({ axes: [{ Not Applicable minorTickLines: { visible: true}]});</p> <p>Property:minorTick Width of minorTickLines Property:minorTickLines.widthlet chart: Chart = new Chart({ axes: [{ minorTickLines: { width: 2} }]);chart.appendTo('#chart');</p> <p>Property:minorTick Height of minorTickLines Property:minorTickLines.heightlet chart: Chart = new Chart({ axes: [{ minorTickLines: { height: 2} }]);chart.appendTo('#chart');</p> <p>Property:minorTick Color of minorTickLines Property:minorTickLines.colorlet chart: Chart = new Chart({ axes: [{ minorTickLines: { color: 'black' } }]);chart.appendTo('#chart');</p> <p>Property:minorTick Opacity of minorTickLines Not Applicable</p> <p>Minor ticks per interval of primary Y Axis</p> <p>Property:minorTick sPerInterval\$("#container").ejChart({ axes: [{ minorTicksPerInterval: 4 }]);chart.appendTo('#chart');</p> <p>Property:name\$("#container").ejChart({ axes: [{ name:</p>
--	--	--

		<pre> prim { name: 'primaryYAxis' aryY 'primaryYAxis' }}}}];chart.appendTo('#chart'); Axis }]]}); Ori Property:orientation ntati n\$("#container" on of).ejChart({ prim axes: [{ Not Applicable aryY orientation: Axis 'Vertical' }]]}); Plot Property:plotOffset offse \$("#container") t for .ejChart({ prim axes: [{ aryY plotOffset: 0 Axis }]]}); mini Property:range.minimum mum \$("#container").ejChart({ for axes: [{ prim range: { aryY minimum: 10 Axis }]]}); maxi Property:range.maximum mum \$("#container").ejChart({ for axes: [{ prim range: { aryY maximum: 10 Axis }]]}); inter Property:range.interval val \$("#container").ejChart({ for axes: [{ prim range: { aryY interval: 1 Axis }]]}); Rang Property:rangePadding ePad \$("#container").ejChart({ ding axes: [{ for rangePadding: prim 'None' aryY }]]}); Axis }]]}); </pre>
--	--	--

		<p>Rounding Places Property:roundingPlaces</p> <pre> Place \$("#container") .ejChart({ axes: [{ roundingPlaces: 3 }] }); </pre> <p>Scrollbar Settings Property:scrollbarSettings</p> <pre> \$("#container") .ejChart({ axes: [{ scrollbarSettings : { } }] }); </pre> <p>Not Applicable</p> <p>Tick Position Property:tickLinesPosition</p> <pre> \$("#container").ejChart({ axes: [{ tickLinesPosition: 'Inside' }] }); </pre> <p>Property:tickPosition</p> <pre> let chart: Chart = new Chart({ axes: [{ tickPosition: 'Inside' }] }); chart.appendTo('#chart'); </pre> <p>Value Type Property:valueType</p> <pre> \$("#container") .ejChart({ axes: [{ valueType: 'DateTime' }] }); </pre> <p>Property:valueType</p> <pre> let chart: Chart = new Chart({ axes: [{ valueType: 'DateTime' }] }); chart.appendTo('#chart'); </pre> <p>Visible Property:visible</p> <pre> \$("#container").ejChart({ axes: [{ visible: true }] }); </pre> <p>Property:visible</p> <pre> let chart: Chart = new Chart({ axes: [{ visible: true }] }); chart.appendTo('#chart'); </pre> <p>Zoom Factor Property:zoomFactor</p> <pre> \$("#container") .ejChart({ axes: [{ zoomFactor: 0.3 }] }); </pre> <p>Property:zoomFactor</p> <pre> let chart: Chart = new Chart({ axes: [{ zoomFactor: 0.3 }] }); chart.appendTo('#chart'); </pre> <p>Zoom Position Property:zoomPosition</p> <pre> \$("#container") .ejChart({ axes: [{ </pre>
--	--	---

		<pre> prim zoomPosition: aryY 0.3 }]]); Axis label Property:labelBord Bord er\$("#container er of ").ejChart({ prim axes: [{ aryY labelBorder: { Axis color: 'red', width: 2} }]]); title Property:title.text\$ of (\$("#container"). prim ejChart({ axes: aryY Chart({ axes: [{ title: 'Chart Axis [{ title: { text: 'Chart title' title' } }]]); title' } }]]); Strip Line Property:stripLine\$ of (\$("#container"). prim ejChart({ axes: aryY Chart({ axes: [{ stripLines: [] Axis [{ stripLine: [] }]]); Multi Property:multiLevel level Labels\$("#contai label ner").ejChart({ s of axes: [{ axes multiLevelLabel s: [] }]]); skele ton for Not Applicable an axes Property:skeletonlet chart: Chart = new Chart({ axes: [{ skeleton: 'yMd' }}]);chart.appendTo('#chart'); Property:skeletonTypelet chart: Chart = new Chart({ axes: [{ skeletonType: 'DateTime' }}]);chart.appendTo('#chart'); skele ton type for Not Applicable an axes ## Rows Beha viour API in Essential JS 1 API in Essential JS 2 rows Property:rowDefinit in ions\$("#chart") chart .ejChart({ Property:rowslet chart: Chart = new Chart({ rows: </pre>
--	--	---

		<pre> rowDefinitions: [];});chart.app [];}); endTo('#chart') ; Property:unit \$("#container") .ejChart({ unit rowDefinitions Not Applicable :[{unit : "percentage"}}]); Property:heightlet height Property:rowHeight chart: Chart = ht of \$("#chart").ejC new Chart({ rows hart({ rows: [{ rowDefinitions: height: in [{ rowHeight: '300'}]);char chart '50%'}]);}); t.appendTo('#ch art'); Property:lineColor, Property:borderle lineWidth\$("#cha = new Chart({ rt").ejChart({ rows: [{ rowDefinitions: height: '300', Line [{ rowHeight: border: { custo '50%', width: 2, miza lineColor: color: tion 'brown', 'brown'}}]);});c lineWidth: hart.appendTo('# 2});}); chart'); ## Series Behav API in Essential JS 1 API in Essential iour JS 2 Property:bearF fillColorlet chart: Property:bearFillColor\$ Chart = new (\$("#chart").ejChar Chart({ bearFi t({ series: llColor [{bearFillColor: [{bearFillC 'red' }]);}); olor: 'red' }]);});chart .appendTo('# chart'); Property:border\$(\$("#c Property:rows hart").ejChart({ let chart: Border series: [{ Chart = new border: { color: Chart({ 'red', width: 2, series: [{ </pre>
--	--	--

		<pre> dashArray: '10, 5' }]];}); border: { color: 'red', width: 2}]];});chart .appendTo('# #chart'); </pre>
	BoxPlotMode	<pre> Property:boxPlotMode \$("#chart").ejChart({ series: [{ boxPlotMode: 'inclusive' }];}); </pre>
	Minimum radius of Bubble series	<pre> Property:minRadius let chart: Chart = new Chart({ series: [{ minRadius: 2 }];});chart .appendTo('# #chart'); </pre>
	Maximum radius of Bubble series	<pre> Property:maxRadius let chart: Chart = new Chart({ series: [{ maxRadius: 2 }];});chart .appendTo('# #chart'); </pre>
	bubbleFillColor	<pre> Property:bubbleFillColor \$("#chart").ejChart({ series: [{bubbleFillColor: 'red' }]];}); </pre>

		<pre> .appendTo('#chart'); Property:cardinalSplineTension onlet chart: Chart = new Chart({ series: [{ cardinalSplineTension: 0.5 }];});chart .appendTo('#chart'); Property:columnWidth let chart: Chart = new Chart({ series: [{ columnWidth: 0.5 }];});chart .appendTo('#chart'); Property:columnSpacing let chart: Chart = new Chart({ series: [{ columnSpacing: 0.5 }];});chart .appendTo('#chart'); Property:cornerRadius.topLeft let chart: Chart = new Chart({ series: [{ topLeft: 0 }];});chart .appendTo('#chart'); Property:cornerRadius.topRight topRight\$("#chart") </pre>
	Cardinal Spline Tension for series	<pre> Property:cardinalSplineTension\$("#chart") .ejChart({ series: [{ cardinalSplineTension: 0.5 }];}); </pre>
	Column Width for rectangle series	<pre> Property:columnWidth\$("#chart").ejChart({ series: [{ columnWidth: 0.5 }];}); </pre>
	Column Spacing for rectangle series	<pre> Property:columnSpacing\$("#chart").ejChart({ series: [{ columnSpacing: 0.5 }];}); </pre>
	Top Left radius for rectangle series	<pre> Property:cornerRadius.topLeft\$("#chart").ejChart({ series: [{ topLeft: 0 }];}); </pre>
	Top Right radius for rectangle series	<pre> Property:cornerRadius.topRight\$("#chart") </pre>

		<pre> radius .ejChart({ for series: [{ rectan topRight: 0 gle }];}); series </pre>	<pre> htlet chart: Chart = new Chart({ series: [{ topRight: 0 }];});chart .appendTo('#chart'); </pre>
		<pre> botto mRigh Property:cornerRadius. t bottomRight\$("#chart radius t").ejChart({ for series: [{ rectan bottomRight: 0 gle }];}); series </pre>	<pre> Property:corner Radius.bottom Rightlet chart: Chart = new Chart({ series: [{ bottomRight : 0 }];});chart .appendTo('#chart'); </pre>
		<pre> botto mLeft Property:cornerRadius. radius bottomLeft\$("#chart for ").ejChart({ rectan series: [{ gle bottomLeft: 0 series }];}); </pre>	<pre> Property:corner Radius.bottom Leftlet chart: Chart = new Chart({ series: [{ bottomLeft: 0 }];});chart .appendTo('#chart'); </pre>
		<pre> DashA Property:dashArray\$(rray #chart").ejChart(prope { series: [{ rty dashArray: '10, 5' }];}); </pre>	<pre> Property:dashA rraylet chart: Chart = new Chart({ series: [{ dashArray: '10, 5' }];});chart .appendTo('#chart'); </pre>
		<pre> DataS Property:dataSource\$(ource "#chart").ejChart for ({ series: [{ series dataSource: [] }];}); </pre>	<pre> Property:dashA rraylet chart: Chart = new Chart({ series: [{ dataSource: </pre>

		<pre> [] });});chart .appendTo('#chart'); Property:draw Typelet chart: chart: Property:drawType\$(" #chart").ejChart(Chart = new Chart({ for { series: [{ series: [{ drawType: 'Line' drawType: 'Line' }];}); }];});chart .appendTo('#chart'); </pre>
Draw type for Polar series	Property:drawType\$(" #chart").ejChart({ series: [{ drawType: 'Line' }];});	Property:emptyPointSettings.visible\$("#chart").ejChart({ series: [{ emptyPointSettings: { visible: false } }];});
Empty Points etting s for series	Property:emptyPointSettings.displayMode\$("#chart").ejChart({ series: [{ displayMode: 'gap' }];});	Property:emptyPointSettings.filllet chart: Chart = new Chart({ series: [{ fill: 'red' }];});chart.appendTo('#chart');
Empty Point Displa Y mode	Property:emptyPointSettings.color\$("#chart").ejChart({ series: [{ color: 'red' }];});	Property:emptyPointSettings.filllet chart: Chart = new Chart({ series: [{ fill: 'red' }];});chart.appendTo('#chart');
Empty Point color	Property:emptyPointSettings.border\$("#chart").ejChart({ series: [{	Property:filllet chart: Chart = new Chart({

		<pre> Border { emptyPointSetting series: [{ s: { color: 'red', width: 2 } }]; } Property:animation.enable let chart: Chart = new Chart({ series: [animation: { enable: false }] }); chart .appendTo('#chart'); Property:animation.duration let chart: Chart = new Chart({ series: [animation: { duration: 1000 }] }); chart .appendTo('#chart'); Property:animation.delay let chart: Chart = new Chart({ series: [animation: { delay: 100 }] }); chart .appendTo('#chart'); Property:dragSettings let chart: Chart = new Chart({ series: [dragSettings: { </pre>
--	--	---

		<pre> mode: 'X' }];}); </pre>
Error bar settings for series	<pre> Property:errorBarSettings\$("#chart").ejChart({ series: [{ errorBarSettings: { } }]}); </pre>	Property:errorBarSettings <pre> let chart: Chart = new Chart({ series: [{errorBarSettings: { } }]}); </pre>
Close series	<pre> Property:isClosed\$("#chart").ejChart({ series: [{ isClosed: true }]}); </pre>	Property:isClosed <pre> let chart: Chart = new Chart({ series: [{ isClosed: true }]}); </pre>
Stacking Property for series	<pre> Property:isStacking\$("#chart").ejChart({ series: [{ isStacking: true }]}); </pre>	Not Applicable
Line cap for series	<pre> Property:lineCap\$("#chart").ejChart({ series: [{ lineCap: 'butt' }]}); </pre>	Not Applicable
Line join for series	<pre> Property:lineCap\$("#chart").ejChart({ series: [{ lineJoin: 'round' }]}); </pre>	Not Applicable
Opacity for series	<pre> Property:opacity\$("#chart").ejChart({ series: [{ opacity: 0.7 }]}); </pre>	Property:errorBarSettings <pre> let chart: Chart = new Chart({ series: [{ opacity: 0.7 }]}); </pre>
Outlier settings	<pre> Property:outLierSettings\$("#chart").ejChart({ series: [{ outLierSettings: { shape: </pre>	Not Applicable

		<pre> gs of 'rectangle' , series size: { height: 30, width: 20}}];}); </pre>
		<pre> Property:point ColorMapping1 et chart: Chart = new Chart({ series: [{ pointColorM apping: 'color' }];});chart .appendTo('#chart'); </pre>
		<pre> Property:palette\$("#c hart").ejChart({ series: [{ palette: "ColorFieldName" }];}); </pre>
		<pre> Property:point ColorMapping1 et chart: Chart = new Chart({ series: [{ pointColorM apping: 'color' }];});chart .appendTo('#chart'); </pre>
		<pre> Property:positiveFill\$ ("#chart").ejChart ({ series: [{ positiveFill: "red" }];}); </pre>
		<pre> Property:point ColorMapping1 et chart: Chart = new Chart({ series: [{ pointColorM apping: 'color' }];});chart .appendTo('#chart'); </pre>
		<pre> Property:showMedian \$("#chart").ejCha rt({ series: [{ showMedian: true }];}); </pre>
		<pre> Property:point ColorMapping1 et chart: Chart = new Chart({ series: [{ showMean: false }];});chart .appendTo('#chart'); </pre>
		<pre> Property:stacki ngGrouplet chart: Chart = new Chart({ series: [{ stackingGro up: 'group' }];});chart .appendTo('#chart'); </pre>
		<pre> Property:stackingGrou p\$("#chart").ejCh art({ series: [{ stackingGroup: 'group' }];}); </pre>

		<p>Specifies the type of the series to render in chart.</p> <p>Property: type <pre>let chart: Chart = new Chart({ series: [{ type: 'Line' }];});chart .appendTo('#chart');</pre></p>
		<p>Defines the visibility of the series.</p> <p>Property: visibility <pre>let chart: Chart = new Chart({ series: [{ visible: true }];});chart .appendTo('#chart');</pre></p>
		<p>Enables or disables the visibility of legend item.</p> <p>Property: toggleVisibility <pre>let chart: Chart = new Chart({ legendSettings: [{ toggleVisibility: true }];});chart .appendTo('#chart');</pre></p>
		<p>Specifies the different types of spline curve.</p> <p>Property: splineType <pre>let chart: Chart = new Chart({ legendSettings: [{ splineType: 'Natural' }];});chart .appendTo('#chart');</pre></p>
		<p>Specifies the name of the x-axis</p> <p>Property: xAxisName <pre>let chart: Chart = new Chart({ series: [{</pre></p>

		<p>that 'secondaryXAxis' xAxisName: has to }};}); 'secondaryX be Axis' associ }};});chart ated .appendTo('' with #chart); this series. Add an axis instan ce with this name to axes collect ion.</p> <p>Name of the prope rty in the datas Property:xName\$("#c ource hart").ejChart({ that series: [{ xName contai : 'x' }]]}); ns x value for the series.</p> <p>Specifi es the name of the y-axis that has to be associ ated with</p>	<p>Property:xNam elet chart: Chart = new Chart({ series: [{ xName: 'x' }]]});chart .appendTo('' #chart);</p> <p>Property:yAxis Namelet chart: Chart = new Chart({ series: [{ yAxisName: 'secondaryY Axis' }]]});chart .appendTo('' #chart);</p>
--	--	--	---

		<p>this series. Add an axis instance with this name to axes collection.</p> <p>Name of the property in the datasources that contains value for the series.</p> <p>Name of the property in the datasources that contains high value for the series.</p>	<pre>Property:yName\$("#chart").ejChart({ series: [{ yName : 'y' }]}); Property:yName let chart: Chart = new Chart({ series: [{ yName: 'y' }]});chart .appendTo('#chart');</pre> <pre>Property:high\$("#chart").ejChart({ series: [{ high : 'y' }]}); Property:high1 let chart: Chart = new Chart({ series: [{ high: 'y' }]});chart .appendTo('#chart');</pre>
--	--	--	--

		<p>Name of the property in the data source that contains low value for the series.</p> <p>Property:low\$("#chart").ejChart({ series: [{ low : 'y' }]});</p> <p>Property:low let chart: Chart = new Chart({ series: [{ low: 'y' }];});chart .appendTo('#chart');</p>
		<p>Name of the property in the data source that contains close value for the series.</p> <p>Property:close\$("#chart").ejChart({ series: [{ close : 'y' }]});</p> <p>Property:close let chart: Chart = new Chart({ series: [{ close: 'y' }];});chart .appendTo('#chart');</p>
		<p>Name of the property in the data source that contains open value for the series.</p> <p>Property:open\$("#chart").ejChart({ series: [{ open : 'y' }]});</p> <p>Property:open let chart: Chart = new Chart({ series: [{ open: 'y' }];});chart .appendTo('#chart');</p>

		<p>Option to add trend lines to chart.</p> <p>Property:trendLines\$ ("#chart").ejChart ({ series: [{ trendLines : [{}}]});});</p> <p>Property:trendLineslet chart: Chart = new Chart({ series: [{ trendLines : [{}}]});});chart.appendTo('#chart');</p> <p>Options for customizing the appearance of the series or data point while highlighting.</p> <p>Property:highlightSettings\$ ("#chart").ejChart ({ series: [{ highlightSettings : {} }]});});</p> <p>Not applicable.</p> <p>Options for customizing the appearance of the series /data point on selection.</p> <p>Property:selectionSettings\$ ("#chart").ejChart ({ series: [{ selectionSettings : {} }]});});</p> <p>Not applicable.</p> <p>## marker</p> <p>Property:visible\$ ("#chart").ejChart ({ series: [{ marker: Property:visiblelet chart: Chart =</p>
--	--	---

		<pre> marke { visible: true } new r];)); Chart({ series: [{ marker: { visible: false } }];});char t.appendTo ('#chart); Property:visibl e let chart: Chart = new Property:fill\$("#chart Fill for ").ejChart({ new Chart({ series: [{ marker: series: [{ { fill : 'red' } marker: { }];}); fill : 'red' } }];});char t.appendTo ('#chart); Property:opac ity let chart: Chart = new Property:opacity\$("#ch Opaci art").ejChart({ new ty for Chart({ serie series: [{ marke { marker: { opacity : 0.5 } marker: { }];}); opacity : 0.5 } }];});char t.appendTo ('#chart); Property:shap e let chart: Chart = new Property:shape\$("#cha Shape rt").ejChart({ new of Chart({ serie series: [{ marke { marker: { shape : 'Circle' marker: { } }];}); shape : 'Triangle' } }];});char t.appendTo ('#chart); </pre>
--	--	---

		<pre> Property:image let chart: Chart = new chart").ejChart({ Chart({ series: [{ marker: series: [{ { imageUrl : '' } marker: { }}]);}); imageUrl : '' } });});char t.appendTo ('#chart); </pre>
Image Url of marker	Property:imageUrl\$ ("# chart").ejChart({ series: [{ marker: { imageUrl : '' } }]);});	<pre> Property:shap let chart: Chart = new Chart({ series: [{ marker: { border : { width: 2, color: 'red' } } }]);}); color: 'red' } } });});char t.appendTo ('#chart); </pre>
Border of marker	Property:border\$ ("#ch art").ejChart({ series: [{ marker: { border : { width: 2, color: 'red' } } }]);});	<pre> Property:heig htlet chart: Chart = new #chart").ejChart({ Chart({ series: [{ marker: series: [{ { size: { height: marker: { 30} } }]);}); height: 25 } });});char t.appendTo ('#chart); </pre>
Height of marker	Property:size.height\$ ("# chart").ejChart({ series: [{ marker: { size: { height: 30} } }]);});	<pre> Property:widt hlet chart: Chart = new chart").ejChart({ Chart = new Chart({ series: [{ marker: series: [{ { size: { width: marker: { 30} } }]);}); width: 25 } } </pre>
Width of marker	Property:size.width\$ ("# chart").ejChart({ series: [{ marker: { size: { width: 30} } }]);});	

		<pre> });});char t.appendTo ('#chart); Property:width hlet chart: Chart = new chart").ejChart({ series: [{ marker: { size: { width: 30} } }];}); });});char t.appendTo ('#chart); Property:mar ker.dataLabel let chart: Chart = new chart").ejCh art({ series: [{ marker: {dataLabel: { } } }];}); });});char t.appendTo ('#chart); Property:data Label.visible et chart: Chart = new chart").ejCha rt({ series: [{ marker: {dataLabel: { visible: true } } }];}); });});char t.appendTo ('#chart); Property:dataLabel.text MappingName\$("#cha rt").ejChart({ series: [{ marker: {dataLabel: { textMappingName: '' } } }];}); });});char t.appendTo ('#chart); </pre>
Width of marker	Property:size.width\$("#chart").ejChart({ series: [{ marker: { size: { width: 30} } }];});	<pre> });});char t.appendTo ('#chart); Property:width hlet chart: Chart = new chart").ejChart({ series: [{ marker: { size: { width: 30} } }];}); });});char t.appendTo ('#chart); Property:mar ker.dataLabel let chart: Chart = new chart").ejCh art({ series: [{ marker: {dataLabel: { } } }];}); });});char t.appendTo ('#chart); Property:data Label.visible et chart: Chart = new chart").ejCha rt({ series: [{ marker: {dataLabel: { visible: true } } }];}); });});char t.appendTo ('#chart); Property:dataLabel.text MappingName\$("#cha rt").ejChart({ series: [{ marker: {dataLabel: { textMappingName: '' } } }];}); });});char t.appendTo ('#chart); </pre>
Data labels setting of marker	Property:marker.dataLabel\$("#chart").ejChart({ series: [{ marker: {dataLabel: { } } }];});	<pre> });});char t.appendTo ('#chart); Property:width hlet chart: Chart = new chart").ejChart({ series: [{ marker: { size: { width: 30} } }];}); });});char t.appendTo ('#chart); Property:mar ker.dataLabel let chart: Chart = new chart").ejCh art({ series: [{ marker: {dataLabel: { } } }];}); });});char t.appendTo ('#chart); Property:data Label.visible et chart: Chart = new chart").ejCha rt({ series: [{ marker: {dataLabel: { visible: true } } }];}); });});char t.appendTo ('#chart); Property:dataLabel.text MappingName\$("#cha rt").ejChart({ series: [{ marker: {dataLabel: { textMappingName: '' } } }];}); });});char t.appendTo ('#chart); </pre>
Visibility of data label	Property:dataLabel.visible\$("#chart").ejChart({ series: [{ marker: {dataLabel: { visible: true } } }];});	<pre> });});char t.appendTo ('#chart); Property:width hlet chart: Chart = new chart").ejChart({ series: [{ marker: { size: { width: 30} } }];}); });});char t.appendTo ('#chart); Property:mar ker.dataLabel let chart: Chart = new chart").ejCh art({ series: [{ marker: {dataLabel: { } } }];}); });});char t.appendTo ('#chart); Property:data Label.visible et chart: Chart = new chart").ejCha rt({ series: [{ marker: {dataLabel: { visible: true } } }];}); });});char t.appendTo ('#chart); Property:dataLabel.text MappingName\$("#cha rt").ejChart({ series: [{ marker: {dataLabel: { textMappingName: '' } } }];}); });});char t.appendTo ('#chart); </pre>
Text mapping name of	Property:dataLabel.text MappingName\$("#chart").ejChart({ series: [{ marker: {dataLabel: {textMappingName: '' } } }];});	<pre> });});char t.appendTo ('#chart); Property:width hlet chart: Chart = new chart").ejChart({ series: [{ marker: { size: { width: 30} } }];}); });});char t.appendTo ('#chart); Property:mar ker.dataLabel let chart: Chart = new chart").ejCh art({ series: [{ marker: {dataLabel: { } } }];}); });});char t.appendTo ('#chart); Property:data Label.visible et chart: Chart = new chart").ejCha rt({ series: [{ marker: {dataLabel: { visible: true } } }];}); });});char t.appendTo ('#chart); Property:dataLabel.text MappingName\$("#cha rt").ejChart({ series: [{ marker: {dataLabel: { textMappingName: '' } } }];}); });});char t.appendTo ('#chart); </pre>

		dataLabel: { name: ' ' } } }]); chart.appendTo('#chart');
		Property: dataLabel.filllet chart: Chart = new Chart({ series: [{ marker: { dataLabel: { fill: 'pink' } } }]); chart.appendTo('#chart');
	Fill color of data label	Property: dataLabel.fill\$ ("#chart").ejChart({ series: [{ marker: { dataLabel: { fill: 'pink' } } }]);
		Property: dataLabel.filllet chart: Chart = new Chart({ series: [{ marker: { dataLabel: { fill: 'pink' } } }]); chart.appendTo('#chart');
		Property: dataLabel.filllet chart: Chart = new Chart({ series: [{ marker: { dataLabel: { opacity: 0.6 } } }]); chart.appendTo('#chart');
	Opacity of data label	Property: dataLabel.opacity\$ ("#chart").ejChart({ series: [{ marker: { dataLabel: { opacity: 0.6 } } }]);
		Property: dataLabel.opacity\$ ("#chart").ejChart({ series: [{ marker: { dataLabel: { opacity: 0.4 } } }]); chart.appendTo('#chart');
		Property: dataLabel.positionlet chart: Chart = new Chart({ series: [{ marker: { dataLabel: { position: 'Top' } } }]); chart.appendTo('#chart');
	Text position of data label	Property: dataLabel.textPosition\$ ("#chart").ejChart({ series: [{ marker: { dataLabel: { textPosition: 'middle' } } }]);
		Property: dataLabel.textPosition\$ ("#chart").ejChart({ series: [{ marker: { dataLabel: { position: 'Top' } } }]); chart.appendTo('#chart');
	Align ment	Property: dataLabel.verticalAlignment\$ ("#chart")
		Property: dataLabel.alignment\$ ("#chart")

		<pre> of t").ejChart({ ntlet data series: [{ marker: chart: label {dataLabel: { Chart = verticalAlignment: new 'near' } } }];}); Chart({ series: [{ marker: { dataLabel: { alignment: 'Near' } } }];});char t.appendTo ('#chart); Property:data Label.alignment ntlet chart: Property:dataLabel.bord er\$("#chart").ejCha rt({ series: [{ marker: {dataLabel: { border: { color: 'blue', width: 2, opacity: 0.4} } } }];}); width: 2 } } } }];});char t.appendTo ('#chart); Property:dataLabel.offse t\$("#chart").ejCha rt({ series: [{ marker: {dataLabel: { offset: { x: 5, y: 6 } } } }];}); Not Applicable Property:data Label.margin1 et chart: Chart = new Chart({ series: [{ marker: { dataLabel: { margin: { top: 10, bottom: </pre>
		<pre> bottom: </pre>

		<pre> 10, left: 10, right: 10 } } } }];});char t.appendTo ('#chart); </pre>
		<pre> Property:data Label.marginl et chart: Chart = new Chart({ series: [{ marker: {dataLabel: { font: { fontFamily: 'SegoeUI', fontStyle: 'italic', fontWeight: '600', opacity: 0.5, size: 12, color: 'red' }} } }];}); </pre>
Font of data label		<pre> Property:dataLabel.bord er\$("#chart").ejCha rt({ series: [{ marker: {dataLabel: { font: { fontFamily: 'SegoeUI', fontStyle: 'italic', fontWeight: '600', opacity: 0.5, size: 12, color: 'red' }} } }];}); </pre>
		<pre> Property:data Label.templat elet chart: Chart = new Chart({ series: [{ marker: {dataLabel: { template: ' Chart ' } } }];});char t.appendTo ('#chart); </pre>
		<pre> Property:dataLabel.tem plate\$("#chart").ej Chart({ series: [{ marker: {dataLabel: { template: ' Chart ' } } }]}); </pre>
HTML templ ate in datal abel		<pre> Property:dataLabel.tem plate\$("#chart").ej Chart({ series: [{ marker: {dataLabel: { template: ' Chart ' } } }]}); </pre>

			<div>Property:data</div> <div>Label.rx</div> <div>let</div> <div>chart:</div> <div>Chart =</div> <div>new</div> <div>Chart({</div> <div>series: [{</div> <div>marker:</div> <div>{dataLabel</div> <div>: { rx: 10</div> <div>} }</div> <div>}]};</div> <div>chart.appendTo</div> <div>('chart');</div>
	<div>Rounded</div> <div>corner</div> <div>radius</div> <div>X</div>	Not Applicable	<div>Property:data</div> <div>Label.ry</div> <div>let</div> <div>chart:</div> <div>Chart =</div> <div>new</div> <div>Chart({</div> <div>series: [{</div> <div>marker:</div> <div>{dataLabel</div> <div>: { ry: 10</div> <div>} }</div> <div>}]};</div> <div>chart.appendTo</div> <div>('chart');</div>
	<div>Rounded</div> <div>corner</div> <div>radius</div> <div>Y</div>	Not Applicable	
	<div>Maximum</div> <div>Label</div> <div>width</div> <div>for</div> <div>data</div> <div>label</div>	<div>Property:dataLabel.maxi</div> <div>mumLabelWidth\$("#ch</div> <div>art").ejChart({</div> <div>series: [{ marker:</div> <div>{dataLabel: {</div> <div>maximumLabelWidth:</div> <div>20}} }</div> <div>}]};</div>	Not Applicable
	<div>Enable</div> <div>wrapping of</div> <div>text</div> <div>for</div> <div>data</div> <div>label</div>	<div>Property:dataLabel.enab</div> <div>leWrap\$("#chart").e</div> <div>jChart({ series:</div> <div>[{ marker:</div> <div>{dataLabel: {</div> <div>enableWrap: true</div> <div>}} }</div> <div>}]};</div>	Not Applicable
	<div>To</div> <div>show</div> <div>contrast</div> <div>color</div>	<div>Property:dataLabel.sho</div> <div>wContrastColor\$("#ch</div> <div>art").ejChart({</div> <div>series: [{ marker:</div> <div>{dataLabel: {</div>	Not Applicable

		<pre> for showContrastColor: data true }} } }];}); label To Property:dataLabel.sho show wEdgeLabels\$("#char edge t").ejChart({ label series: [{ marker: for {dataLabel: { data showEdgeLabels: label true }} } }];}); ## TrendLines Be hav API in Essential JS 1 API in Essential JS 2 iou r Tre Property:series.tre ndli ndLineslet nes Property:series.trendL chart: Chart = set ines\$("#chart").e new Chart({ tin jChart({ series: series: [{ gs [{ trendLines: trendLines: []}}]); []}}]);chart.a ppendTo('#char t'); Visi Property:trendLines.vi bili sibility\$("#chart") . ty ejChart({ of series: [{ tre trendLines: [ndli visibility: true ne]}}]); Typ Property:trendLines.ty e s.typelet chart: of Chart = new tre Chart({ ndli series: [{ ine trendLines: [type: 'Polynomial']]]});chart.appe ndTo('#chart') ; Na Property:trendLines.n Property:trendLine me ame\$("#chart").e s.namelet jChart({ series: chart: Chart = </pre>
--	--	--

		<pre> of [{ trendLines: [new Chart({ tre name: series: [{ ndL 'trendLine' trendLines: [ine]}}]); name: 'trendLine']] });chart.append dTo('#chart'); Property:trendLine Per Property:trendLines.p s.periodlet iod eriod\$("#chart"). chart: Chart = of ejChart({ new Chart({ tre series: [{ series: [{ ndL trendLines: [trendLines: [ine period: 45 period: 45]]]]});chart. appendTo('#cha rt'); Pol yno mia l ord er Property:trendLines.p s.polynomialOrder for olynomialOrder\$("#c let chart: Pol hart").ejChart({ Chart = new yno series: [{ Chart({ mia trendLines: [series: [{ l polynomialOrder: trendLines: [typ 3]}}]); polynomialOrde e r: tre 3]}}]); 3]]}}]);chart.a ndL ppendTo('#char ine t'); s Bac kw ard Property:trendLines.b s.backwardforecast for ackwardforecast\$("# let chart: eco chart").ejChart(Chart = new st { series: [{ Chart({ for trendLines: [series: [{ tre backwardforecast trendLines: [ndL : 3]}}]); backwardforeca ine st: s 3]]}}]);chart.a ppendTo('#char t'); </pre>
--	--	--

		<pre> For wa rd rd Property:trendLines.f for orwardForecast\$("#c eco hart").ejChart({ st series: [{ for trendLines: [tre forwardForecast: ndL 3]}]}}); ine s </pre>	<pre> Property:trendLine s.forwardForecast1 et chart: Chart = new Chart({ series: [{ trendLines: [forwardForecas t: 3]}]}});chart.a ppendTo('#char t'); </pre>
		<pre> Fill Property:trendLines.fil for l\$("#chart").ejCh tre art({ series: [{ ndL trendLines: [ine fill: '#EEFFCC' s]}]}}); </pre>	<pre> Property:trendLine s.filllet chart: Chart = new Chart({ series: [{ trendLines: [fill: 'EEFFCC']}]}}); chart.appendTo ('#chart'); </pre>
		<pre> Wi dth Property:trendLines.w for idth\$("#chart").e tre jChart({ series: ndL [{ trendLines: [ine width: 2]}]}}); s </pre>	<pre> Property:trendLine s.widthlet chart: Chart = new Chart({ series: [{ trendLines: [width: 2]}]}});chart.a ppendTo('#char t'); </pre>
		<pre> Int erc ept Property:trendLines.in val tercept\$("#chart") ue .ejChart({ for series: [{ tre trendLines: [ndL intercept: 2 ine]}]}}); s </pre>	<pre> Property:trendLine s.interceptlet chart: Chart = new Chart({ series: [{ trendLines: [intercept: 2]}]}});chart.a ppendTo('#char t'); </pre>
		<pre> Leg en d Not Applicable sha pe </pre>	<pre> Property:trendLine s.legendShapelet chart: Chart = new Chart({ series: [{ trendLines: [</pre>

		<pre> for tre ndL ine s Ani ma tio n set tin gs for tre ndL ine s Ma rke r set tin gs for tre ndL ine s To olti p for tre ndL ine s Das hAr ray for tre ndL </pre>	<pre> legendShape: 'Rectangle'[]}] });chart.append dTo('#chart'); Property:trendLine s.animationlet chart: Chart = new Chart({ series: [{ trendLines: [animation: { enable: true }]]]);chart.a ppendTo('#char t'); Property:trendLine s.markerlet chart: Chart = new Chart({ series: [{ trendLines: [{marker: { visible: true }]]]]]);chart. appendTo('#cha rt'); Property:trendLine s.enableTooltiplet chart: Chart = new Chart({ series: [{ trendLines: [{enableTooltip : true }]]]]]);chart.a ppendTo('#char t'); Property:trendLines.d ashArray\$("#chart").ejChart({ series: [{ trendLines: [{ dashArray: '10, 5' }]]]]); </pre>	<pre> Not Applicable Not Applicable Not Applicable. </pre>
--	--	--	--	--

		<pre> ine s Visi ble on Property:trendLines.vi leg sibleOnLegend\$("#c en hart").ejChart({ d series: [{ for trendLines: [{ tre visibleOnLegend: ndL true }]}})); ine s ## Striplines Beh avio API in Essential JS 1 API in Essential JS 2 ur Defa Property:primaryX ult Property:primaryXAxis.striplineslet beh is.striplines\$("#cha new Chart({ avio rt").ejChart({ primaryXAxis: ur primaryXAxis: { { striplines: for stripLines: [{ [{ visible: strip visible: true true lines }]]}); chart.ap pendTo('#chart '); Property:striplines .borderlet chart: Chart = new Chart({ primaryXAxis: { striplines: [{ border: { color: 'red', width: 2} }}]);chart.ap pendTo('#chart '); Back Property:striplines.c Property:striplines grou color\$("#chart").e .borderlet nd jChart({ chart: Chart = colo primaryXAxis: { new Chart({ r for stripLines: [{ primaryXAxis: { striplines: </pre>
--	--	--

		<pre> strip color: 'pink' [{ color: line }]]]]]; 'red']]]]]]; cha rt.appendTo('# chart'); Property:stripLines .startlet chart: Star Property:stripLines.st Chart = new t art\$("#chart").e Chart({ valu jChart({ primaryXAxis: e for primaryXAxis: { { stripLines: strip stripLines: [{ [{ start: line start: 10 }]]]]]; 5]]]]]]]; chart.a ppendTo('#char t'); Property:stripLines .endlet chart: End Property:stripLines.e Chart = new valu nd\$("#chart").ej Chart({ e for Chart({ primaryXAxis: strip primaryXAxis: { { stripLines: line stripLines: [{ [{ end: end: 10 }]]]]]; 5]]]]]]]; chart.a ppendTo('#char t'); Property:stripLines .startFromAxislet Star Property:stripLines.st chart: Chart = tfro artFromAxis\$("#cha new Chart({ mAx rt").ejChart({ primaryXAxis: is primaryXAxis: { { stripLines: for stripLines: [{ [{ strip startFromAxis: startFromAxis: line true }]]]]]; true }]]]]]; char t.appendTo('#c hart'); Property:stripLines .textlet chart: Text Property:stripLines.t Chart = new in ext\$("#chart").e Chart({ strip jChart({ primaryXAxis: line primaryXAxis: { primaryXAxis: stripLines: [{ { stripLines: text: [{ text: 'StripLine; 'stripline']]] }]]]]]; }); chart.appen }]]]]]; dTo('#chart'); Property:stripLines .horizontalAlignme Text Property:stripLines.t Property:stripLines align extAlignment\$("#ch .horizontalAlignme men art").ejChart({ ntlet chart: t in primaryXAxis: { Chart = new </pre>
--	--	--

		<pre> strip stripLines: [{ Chart({ line textAlignment: primaryXAxis: 'Far; }}}}); { stripLines: [{ horizontalAlign ment: 'Far'}}]);cha rt.appendTo('# chart'); Property:stripLines .verticalAlignment let chart: Chart = new Chart({ primaryXAxis: { stripLines: [{ verticalAlignm ent: 'Far'}}]);cha rt.appendTo('# chart'); Property:stripLines .sizelet chart: Chart = new Chart({ primaryXAxis: { stripLines: [{ size: 10 }}]);chart.ap pendTo('#chart '); Property:stripLines .sizelet chart: Chart = new Chart({ primaryXAxis: { stripLines: [{ zIndex: 'Behind' 'Behind' }}]);chart.ap pendTo('#chart '); Property:stripLines.f ontStyle\$("#chart ").ejChart({ primaryXAxis: { stripLines: [{ fontStyle: {} }}]); Property:stripLines .textStylelet chart: Chart = new Chart({ primaryXAxis: { stripLines: [{ textStyle: </pre>
--	--	--

				<pre>{ } }}}});chart.ap pendTo('#chart ');</pre>
			## Multilevel Labels	
	Behavio	ur	API in Essential JS 1	API in Essential JS 2
	Default	Property:primaryX		Axis.multilevelLab
	ult	Property:primaryX		elslet chart:
	beh	s.multilevelLabels\$ ("		Chart = new
	avio	#chart").ejChart		Chart({
	ur	({ primaryXAxis:		primaryXAxis:
	for	{		{
	mult	multilevelLabels		multilevelLab
	ileve	: [{ visible:		els: [{
	lLab	true }]]]]);		}}]]];chart.a
	els			ppendTo('#cha
				rt');
	Default	Property:primaryX		Axis.multilevelLab
	ult	Property:primaryX		elslet chart:
	beh	s.multilevelLabels\$ ("		Chart = new
	avio	#chart").ejChart		Chart({
	ur	({ primaryXAxis:		primaryXAxis:
	for	{		{
	mult	multilevelLabels		multilevelLab
	ileve	: [{ visible:		els: [{
	lLab	true }]]]]);		}}]]];chart.a
	els			ppendTo('#cha
				rt');
				Property:multilev
				elLabels.alignment
	Text	Property:multiLevelLa		tlet chart:
	align	bels.textAlignment\$ (Chart = new
	men	"#chart").ejChar		Chart({
	t for	t({		primaryXAxis:
	mult	primaryXAxis: {		{
	ileve	multilevelLabels		multilevelLab
	lLab	: [{		els: [
	els	textAlignment:		{alignment:
		'Near' }]]]]);		'Near'
				}}]]];chart.a
				ppendTo('#cha
				rt');
	Text	Property:multiLevelLa	Property:multiLev	
	over	bels.textOverflow\$ ("	elLabels.overFlow	

		<pre> flow #chart").ejChart let chart: for ({ primaryXAxis: Chart = new mult { Chart({ multilevelLabels primaryXAxis: ileve : [{ lLab textOverFlow: multilevelLab els 'Trim' }]]}); els: [{overFlow: 'Trim' }]]});chart.a ppendTo('#cha rt'); Property:multiLev elLabels.borderle t chart: Chart = new Chart({ Property:multiLevelLa els.border\$("#char t").ejChart({ for primaryXAxis: { multilevelLab mult multilevelLabels els: [{ ileve : [{ border: { border: { lLab width: 2, color: width: 2, els 'red', type: color: 'red', 'brace' } }]]}); type: 'brace' } }]]});chart.a ppendTo('#cha rt'); Property:multiLev elLabels.categorie s.startlet chart: Chart = new Chart({ Property:multiLevelLa els.start\$("#chart ").ejChart({ primaryXAxis: { multilevelLab multilevelLabels els: [{ : [{ start: 45 categories: [} }]]}); { start: 45}] } }]]});chart.a ppendTo('#cha rt'); Property:multiLevelLa els.start\$("#chart ").ejChart({ primaryXAxis: { multilevelLab multilevelLabels : [{ end: 45 } : [{ end: 45 } }]]}); Property:multiLev elLabels.categorie s.endlet chart: Chart = new Chart({ primaryXAxis: { </pre>
--	--	--

		<pre> multilevelLabels: [{ categories: [{ end: 45}] }] } } } } } ; chart.appendTo('#chart'); Property:multiLevelLabels.categories.text let chart: Chart = new Chart({ primaryXAxis: { multilevelLabels: [{ categories: [{ text: 'text' }] } }] } } } } } ; chart.appendTo('#chart'); Property:multiLevelLabels.categories.maximumTextWidth let chart: Chart = new Chart({ primaryXAxis: { multilevelLabels: [{ categories: [{ maximumTextWidth: 10 }] } }] } } } } } ; chart.appendTo('#chart'); Property:multiLevelLabels.level let chart: Chart = new Chart({ primaryXAxis: { multilevelLabels: [{ level: 2 } }] } } } } } ; ## Methods </pre>
--	--	--

		<p>Behavior API in Essential JS 1 API in Essential JS 2</p> <p>animation Property:chart.animate\$("#chart").ejChart({ animate: () { }}); Not applicable</p> <p>Redraw Property:chart.redraw\$("#chart").ejChart({ redraw: () { }}); Property:chart.refresh() let chart: Chart = new Chart({});chart.appendTo('#chart');chart.width = '400';chart.refresh();</p> <p>Export Property:chart.export\$("#chart").ejChart({ export: () { }}); Property:chart.export() let chart: Chart = new Chart({});chart.export('JPEG', 'chart');chart.appendTo('#chart');</p> <p>Print Property:chart.print\$("#chart").ejChart({ print: () { }}); Property:chart.print() let chart: Chart = new Chart({});chart.print('chart');chart.appendTo('#chart');</p> <p>Add Series Not Applicable Property:chart.addSeries() let chart: Chart = new Chart({});chart.appendTo('#chart');chart.addSeries();</p> <p>Remove Series Not Applicable Property:chart.removeSeries() let chart: Chart = new Chart({});chart.appendTo('#chart');chart.removeSeries();</p> <p>## Events</p>
--	--	--

		<div> <div>Behavio ur</div> <div>API in Essential JS 1</div> </div> <div> <div>API in Essential JS 2</div> </div>
		<div> <div>Fires on annotation click</div> <div>Property:annotationClick</div> <div>\$("#chart").ejChart({</div> <div>annotationClick:</div> <div>() { }));</div> </div> <div>Not applicable</div>
		<div> <div>Fires after animation</div> <div>Property:animationComplete</div> <div>let chart: Chart = new Chart({</div> <div>animationComplete:</div> <div>() { }));</div> </div> <div> <div>Property:animationComplete</div> <div>()let chart: Chart = new Chart({</div> <div>animationComplete:</div> <div>mplete: () => {</div> <div>}});chart.appendTo('#chart');</div> </div>
		<div> <div>Fires on axis label click</div> <div>Property:axisLabelClick</div> <div>\$("#chart").ejChart({</div> <div>axisLabelClick:</div> <div>() { }));</div> </div> <div>Not applicable</div>
		<div> <div>Fires before axis label rendering</div> <div>Property:axisLabelRendering</div> <div>ring\$(\$("#chart").ejChart({</div> <div>axisLabelRendering:</div> <div>: () { }));</div> </div> <div> <div>Property:axisLabelRendering</div> <div>render()let chart: Chart = new Chart({</div> <div>axisLabelRendering:</div> <div>nder: () => {</div> <div>}});chart.appendTo('#chart');</div> </div>
		<div> <div>Fires on axis label mouse move</div> <div>Property:axisLabelMouseMove</div> <div>Move\$(\$("#chart").ejChart({</div> <div>axisLabelMouseMove:</div> <div>: () { }));</div> </div> <div>Not applicable</div>
		<div> <div>Fires on axis label initialize</div> <div>Property:axisLabelInitialize</div> <div>ze\$(\$("#chart").ejChart</div> </div> <div>Not applicable</div>

		<pre> axis rt({ label axisLabelInitializ initi e: () { }); alize Fires befo re Property:axesRangeCalcu axis late\$("#chart").ejCh rang art({ e axesRangeCalculate calc : () { }); ulati on Fires on Property:axisTitleRenderi axis ng\$("#chart").ejCha title rt({ rend axisTitleRendering erin : () { }); g Fires on after Property:afterResize\$(char "#chart").ejChart({ t afterResize: () { resiz }); e Fires on befo Property:beforeResize\$(re "#chart").ejChart(char { beforeResize: () t { }); resiz e Fires on Property:chartClick\$(char "#chart").ejChart({ t chartClick: () { click }); </pre>	<pre> Property:axisRangeCalculated ()let chart: Chart = new Chart({ axisRangeCa lculated: () => { });chart.a ppendTo('#c hart'); Not applicable Not applicable Property:resize dlet chart: Chart = new Chart({ resized: () => { });chart.a ppendTo('#c hart'); Property:chart MouseClicked t chart: Chart = new Chart({ chartMouseC lick: () => { </pre>
--	--	---	---

		<pre> });chart.a ppendTo('#c hart'); </pre>
	<pre> Fires on char Property:chartMouseMo t ve\$("#chart").ejCha rt({ mou chartMouseMove: () se { }); mov e </pre>	<pre> Property:chart MouseMovele t chart: Chart = new Chart({ chartMouseM ove: () => { }});chart.a ppendTo('#c hart'); </pre>
	<pre> Fires on char Property:chartMouseLea t ve\$("#chart").ejCha rt({ mou chartMouseLeave: se () { }); leav e </pre>	<pre> Property:chart MouseLeavele t chart: Chart = new Chart({ chartMouseL eave: () => { }});chart.a ppendTo('#c hart'); </pre>
	<pre> Fires on befo Property:chartDoubleClic re k\$("#chart").ejChar char t({ t chartDoubleClick: dou () { }); ble click </pre>	<pre> Not applicable </pre>
	<pre> Fires on char t Not Applicable mou se up </pre>	<pre> Property:chart mouseUplet chart: Chart = new Chart({ chartmouseU p: () => { }});chart.a ppendTo('#c hart'); </pre>
	<pre> Fires on char Not Applicable t </pre>	<pre> Property:chart mouseDownle t chart: Chart = new </pre>

		<p>mouse down event</p> <p>Fires during the calculation of chart area bounds.</p> <p>You can use this event to customize the bounds of chart area</p> <p>Fires when the dragging is started</p>	<pre>Chart({ chartMouseDown: () => { }});chart.appendTo('#chart');</pre> <p>Property:chartAreaBoundsCalculate\$("#chart").ejChart({chartAreaBoundsCalculate: () { }});</p> <p>Not applicable</p> <p>Property:dragStart\$("#chart").ejChart({dragStart: () { }});</p> <p>Not applicable</p>
--	--	---	---

		<pre> Fires while dragging Property:dragging\$("#chart").ejChart({ dragging: () { // Not applicable } }); </pre>
		<pre> Fires when the dragging is completed Property:dragComplete\$("#chart").ejChart({ dragComplete: () => { // Property:dragComplete Chart = new Chart({ dragComplete: () => { // Chart = new Chart({ // dragComplete: () => { // });chart.appendTo('#chart'); } });chart.appendTo('#chart'); } }); </pre>
		<pre> Fires when chart is destroyed Property:destroy\$("#chart").ejChart({ destroy: () { // Not applicable } }); </pre>
		<pre> Fires after chart is created Property:create\$("#chart").ejChart({ create: () { // Property:loaded dlet chart: Chart = new Chart({ loaded: () => { // Chart = new Chart({ // loaded: () => { // });chart.appendTo('#chart'); } });chart.appendTo('#chart'); } }); </pre>
		<pre> Fires before rendering the data Property:displayTextRendering\$("#chart").ejChart({ displayTextRendering: () { // Property:textRenderer let chart: Chart = new Chart({ textRenderer: () => { // Chart = new Chart({ // textRenderer: () => { // });chart.appendTo('#chart'); } });chart.appendTo('#chart'); } }); </pre>

		<p>label s.</p> <p>Fires , when Property:errorBarRender ing\$("#chart").ejCh art({ errorBarRendering: () { }));</p> <p>Not applicable</p> <p>g.</p> <p>Fires during the calculation of legend bounds.</p> <p>Property:legendBoundsC alculate\$("#chart").e jChart({ legendBoundsCalcul ate: () { }));</p> <p>Not applicable</p> <p>Fires on clicking the legend item .</p> <p>Property:legendItemClick \$("#chart").ejChar t({ legendItemClick: () { }));</p> <p>Not applicable</p> <p>Fires when moving mouse over lege</p> <p>Property:legendItemMo useMove\$("#chart"). ejChart({ legendItemMouseMov e: () { }));</p> <p>Not applicable</p>
--	--	--

		<pre> nd item Fires before render Property:legendItemRender let chart: Chart = new Chart({ legendRender: () => { }});chart.appendTo('#chart'); Fires before render Property:load let chart: Chart = new Chart({ load: () => { }});chart.appendTo('#chart'); Fires when multiple Property:multiLevelLabel let chart: Chart = new Chart({ axisMultiLevelLabelRender: () => { }});chart.appendTo('#chart'); Fires on click Property:pointRegionClick let chart: Chart = new Chart({ pointClick: () => { }});chart.appendTo('#chart'); </pre>
--	--	---

		<pre> Fires when mouse is moved over a point. Property:pointMove let chart: Chart = new Chart({ pointMove : () => { }});chart.a ppendTo('#c hart'); </pre>
		<pre> Fires before rendering a chart. Property:preRender\$ ("#chart").ejChart({ preRender: () { }}); </pre> <p>Not applicable</p>
		<pre> Fires when a point is rendered. Property:pointRender let chart: Chart = new Chart({ pointRender : () => { }});chart.a ppendTo('#c hart'); </pre> <p>Not Applicable</p>
		<pre> Fires after selecting the data in a chart. Property:rangeSelected\$ ("#chart").ejChart ({ rangeSelected: () { }}); </pre> <p>Not applicable</p>
		<pre> Fires after selecting a series. Property:seriesRegionClick\$ ("#chart").ejCha rt({ seriesRegionClick: () { }}); </pre> <p>Not applicable</p>

		<pre> Fires befo re re Property:seriesRendering rend \$("#chart").ejChar erin t({ ga seriesRendering: serie () { }); s. </pre>	<pre> Property:series Render let chart: Chart = new Chart({ seriesRende r : () => { }});chart.a ppendTo('#c hart'); </pre>
		<pre> Fires befo re re rend Property:symbolRenderi erin ng\$("#chart").ejCha g rt({ the symbolRendering: mar () { }); ker sym bols. </pre>	<pre> Not applicable </pre>
		<pre> Fires befo re re Property:trendlineRende rend ring\$("#chart").ejCh erin art({ g trendlineRendering the : () { }); tren dline </pre>	<pre> Not applicable </pre>
		<pre> Fires befo re re rend Property:titleRendering\$ erin ("#chart").ejChart g ({ titleRendering: the () { }); Char t title. </pre>	<pre> Not applicable </pre>
		<pre> Fires Property:subTitleRenderi befo ng\$("#chart").ejCha re rt({ rend subTitleRendering: erin () { }); </pre>	<pre> Not applicable </pre>

		<pre> g the Char t sub title. Fires befo re rend Property:toolTipInitialize erin t({ g toolTipInitialize: the () { }); toolt ip. Fires befo re rend Property:trackAxisToolTi erin p\$("#chart").ejChar g t({ cross trackAxisToolTip: shair () { }); toolt ip in axis Fires befo re rend Property:trackToolTip\$(erin "#chart").ejChart(g { trackToolTip: () trac { }); kball toolt ip. Even t trigg Property:scrollStart\$("# ered chart").ejChart({ when scrollStart: () { n }); scrol l </pre>
		<pre> Property:toolTipRender let chart: Chart = new Chart({ tooltipRender : () => { }});chart.a ppendTo('#c hart'); Not applicable Not applicable Property:scrollStart let chart: Chart = new Chart({ scrollStart : () => { }});chart.a </pre>

		<pre> start s. ppendTo('#c hart'); Even t trigg ered Property:scroll Endlet chart: when hart").ejChart({ Chart = new n scrollEnd: () { Chart({ scrol }}}); scrollEnd: () => { }}};chart.a ends ppendTo('#c . hart'); Even t trigg ered Property:scroll Changelet chart: when "#chart").ejChart(Chart = new n { scrollChange: () scrollChang scrol { }}); e: () => { }}};chart.a chan ppendTo('#c ges. hart'); Fires while e performi ng Property:zoomComplete rect \$("#chart").ejChar angl t({ zoomComplete: e () { }}); zoo min g in char t. Property:zoom Completelet chart: Chart = new Chart({ zoomComple e: () => { }}};chart.a ppendTo('#c hart'); ## Chart properties Behavi API in Essential JS our 1 API in Essential JS 2 selecte Property:selectedDataPointIndexes Property:selectedDataIndexeslet d data :\$("#chart"). chart: Chart = index ejChart({ new selectedDataP Chart({selected DataIndexes: [</pre>
--	--	---

		<pre> ointIndexes: { series: 0, [{ point: seriesIndex: 1}}]);chart.append 0, endTo('#chart') pointIndex: ; 1}}]); </pre>
	<p>sideBy SideSer iesPlac ement for column n based series</p>	<p>Property:sideBySide Placementlet chart: Chart = new Chart({ sideBySidePlace ment: true});chart.ap pendTo('#chart');</p>
	<p>ZoomS ettings</p>	<p>Property:zoomin g:\$("#chart") .ejChart({ zooming: { enable: true, enabledeferre dZoom: true, enablePinch: true, enableMouseWh eel: true, enableScrollBa r: true, toolBarItems: [], type: 'X' });</p> <p>Property:zoomSetti ngslet chart: Chart = new Chart({ zoomSettings: { enable: true, enablePinchZoom ing: true, enableDeferredZ ooming: true enableMouseWhee lZooming: true, enableSelection Zooming: true, enableScrollBar : true });chart.append To('#chart');</p>
	<p>Backgr ound color of the chart</p>	<p>Property:backgro und \$("#container ").ejChart({ background: 'transparent' });</p> <p>Property:backgroun dlet chart: Chart = new Chart({ background: '#EEFFCC'});cha rt.appendTo('#c hart');</p>
	<p>URL of the image to be used as chart</p>	<p>Property:backGro undImageUrl \$("#container ").ejChart({ backGroundIma geUrl : '../images/ch</p> <p>Not Applicable</p>

		<pre> backgr art/wheat.png ound. '}}); Property:border Property:borderlet Custo \$("#container chart: Chart = mizing ").ejChart({ new Chart({ border border: { border: { width: 2, width: 2, color: color: '#CCEEFF', '#CCEEFF'}}});ch chart opacity: art.appendTo('# 0.5}}); chart'); This Property:export()le provid Property:exportS t chart: Chart es ettings\$("#cont = new Chart({ option ainer").ejCha border: { s for rt({ width: 2, custom exportSetting color: izing s: { filename '#CCEEFF'}});ch export : "chart", art.appendTo('# setting angle: '45' chart');chart.e s }}}; xport(type, fileName); Property:chartArea ChartA let chart: rea Chart = new custom { chartArea: { ization , border: { Chart({ background: chartArea: { 'transparent', background: 'transparent' 'transparent', border: { width: 2, color: '#CCEEFF' }}});chart.appen dTo('#chart');c hart.export(typ e, fileName); </pre>
--	--	--

<tr>

<td>Behaviour</td>

<td>API in Essential JS 1</td>

<td>API in Essential JS 2</td>

</tr>

<tr>

<td>Alternate grid band</td>

```

<td>
<b>Property</b>:<i>alternateGridBand </i>
</br>
</br>
<code>
$( "#container" ).ejChart({
primaryXAxis: { alternateGridBand: { even: { fill: 'red' }}}
});

```

```

</code>

```

```

</td>

```

```

<td>

```

```

Not applicable

```

```

</td>

```

```

</tr>

```

```

<tr>

```

```

<td><b>Axis line cross value</b></td>

```

```

<td>

```

```

<b>Property</b>:<i>crossesAt</i>

```

```

</br>

```

```

</br>

```

```

<code>

```

```

$( "#container" ).ejChart({
primaryXAxis: { crossesAt: 0 }
});

```

```

</code>

```

```

</td>

```

```

<td>

```

```

<b>Property</b>:<i>crossesAt</i>

```

```

</br>

```

```

</br>

```

```

<code>

```

```

let chart: Chart = new Chart({
primaryXAxis: { crossesAt: 4}

```



```

});
chart.appendTo('#chart');
</code>
</td>
</tr>
<tr>
<td><b>axis name with which the axis line has to be crossed</b></td>
<td>
<b>Property</b>:<i>crossesInAxis</i>
</br>
</br>
<code>
$("#container").ejChart({
primaryXAxis: { crossesInAxis: " }
});
</code>
</td>
<td>
<b>Property</b>:<i>crossesInAxis</i>
</br>
</br>
<code>
let chart: Chart = new Chart({
primaryXAxis: { crossesInAxis: " }
});
chart.appendTo('#chart');
</code>
</td>
</tr>
<tr>
<td><b>axis elements placed with axis line</b></td>
<td>
<b>Property</b>:<i>showNextToAxisLine </i>

```

</br>

</br>

<code>

```
$("#container").ejChart({
  primaryXAxis: { showNextToAxisLine : true }
});
```

</code>

</td>

<td>

Property:<i>placeNextToAxisLine</i>

</br>

</br>

<code>

```
let chart: Chart = new Chart({
  primaryXAxis: { placeNextToAxisLine: " " }
});
```

```
chart.appendTo('#chart');
```

</code>

</td>

</tr>

<tr>

<td>axis line style</td>

<td>

Property:<i>axisLine.color</i>

</br>

</br>

<code>

```
$("#container").ejChart({
  primaryXAxis: { axisLine: { color : 'red' } }
});
```

</code>

</td>

<td>

```

<b>Property</b>:<i>lineStyle.color</i>
</br>
</br>
<code>
let chart: Chart = new Chart({
primaryXAxis: { lineStyle: { color: 'black' } }
});
chart.appendTo('#chart');
</code>
</td>
</tr>
<tr>
<td><b>axis line dashArray</b></td>
<td>
<b>Property</b>:<i>axisLine.color</i>
</br>
</br>
<code>
$("#container").ejChart({
primaryXAxis: { axisLine: { dashArray : '10, 5' } }
});
</code>
</td>
<td>
<b>Property</b>:<i>lineStyle.dashArray</i>
</br>
</br>
<code>
let chart: Chart = new Chart({
primaryXAxis: { lineStyle: { dashArray: '10, 5' } }
});
chart.appendTo('#chart');
</code>

```

```
</td>
</tr>
<tr>
<td><b>Offset for axis</b></td>
<td>
<b>Property</b><i>axisLine.offset</i>
<br>
<br>
<code>
$( "#container").ejChart({
primaryXAxis: { axisLine: { offset : 10 } }
});
</code>
</td>
<td>
<b>Property</b><i>plotOffset</i>
<br>
<br>
<code>
let chart: Chart = new Chart({
primaryXAxis: { plotOffset: 10 }
});
chart.appendTo('#chart');
</code>
</td>
</tr>
<tr>
<td><b>Visible of an axis</b></td>
<td>
<b>Property</b><i>axisLine.offset</i>
<br>
<br>
<code>
```

```
$("#container").ejChart({
  primaryXAxis: { axisLine: { visible : false } }
});
```

```
</code>
```

```
</td>
```

```
<td>
```

```
<b>Property</b>:<i>visible</i>
```

```
</br>
```

```
</br>
```

```
<code>
```

```
let chart: Chart = new Chart({
  primaryXAxis: { visible: false }
});
```

```
chart.appendTo('#chart');
```

```
</code>
```

```
</td>
```

```
</tr>
```

```
<tr>
```

```
<td><b>Width of an axis</b></td>
```

```
<td>
```

```
<b>Property</b>:<i>axisLine.width</i>
```

```
</br>
```

```
</br>
```

```
<code>
```

```
$("#container").ejChart({
  primaryXAxis: { axisLine: { width : 2 } }
});
```

```
</code>
```

```
</td>
```

```
<td>
```

```
<b>Property</b>:<i>lineStyle.width</i>
```

```
</br>
```

```
</br>
```

```
<code>
```

```
let chart: Chart = new Chart({  
  primaryXAxis: { lineStyle: { width: 3 } }  
});  
chart.appendTo('#chart');
```

```
</code>
```

```
</td>
```

```
</tr>
```

```
<tr>
```

```
<td><b>Column index of an axis</b></td>
```

```
<td>
```

```
<b>Property</b>:<i>columnIndex</i>
```

```
</br>
```

```
</br>
```

```
<code>
```

```
$("#container").ejChart({  
  primaryXAxis: { columnIndex: 2 }  
});
```

```
</code>
```

```
</td>
```

```
<td>
```

```
<b>Property</b>:<i>columnIndex</i>
```

```
</br>
```

```
</br>
```

```
<code>
```

```
let chart: Chart = new Chart({  
  primaryXAxis: { columnIndex: 2 }  
});  
chart.appendTo('#chart');
```

```
</code>
```

```
</td>
```

```
</tr>
```

```
<tr>
```

<td>span of an axis to place horizontally or vertically</td>

<td>

Property:<i>columnSpan</i>

</br>

</br>

<code>

```
$("#container").ejChart({  
  primaryXAxis: { columnIndex: 2 }  
});
```

</code>

</td>

<td>

Property:<i>span</i>

</br>

</br>

<code>

```
let chart: Chart = new Chart({  
  primaryXAxis: { span: 2 }  
});  
chart.appendTo('#chart');
```

</code>

</td>

</tr>

<tr>

<td>Crosshair label of an axis</td>

<td>

Property:<i>crossHairLabel.visible</i>

</br>

</br>

<code>

```
$("#container").ejChart({  
  primaryXAxis: { crossHairLabel: { visible: true } }  
});
```

</code>

</td>

<td>

Property:<i>crossHairTooltip.enable</i>

</br>

</br>

<code>

```
let chart: Chart = new Chart({
  primaryXAxis: { crossHairTooltip: { enable: true }}
});
```

```
chart.appendTo('#chart');
```

</code>

</td>

</tr>

<tr>

<td>Crosshair label color of an axis</td>

<td>

Not applicable

</td>

<td>

Property:<i>crossHairTooltip.fill</i>

</br>

</br>

<code>

```
let chart: Chart = new Chart({
  primaryXAxis: { crossHairTooltip: { fill: 'red' }}
});
```

```
chart.appendTo('#chart');
```

</code>

</td>

</tr>

<tr>

<td>Crosshair label text style</td>

<td>

Not applicable

</td>

<td>

Property:<i>crossHairTooltip.textStyle</i>

</br>

</br>

<code>

```
let chart: Chart = new Chart({
  primaryXAxis: { crossHairTooltip: { textStyle: { } } }
});
```

```
chart.appendTo('#chart');
```

</code>

</td>

</tr>

<tr>

<td>Desired interval count for primaryXAxis</td>

<td>

Property:<i>desiredIntervals</i>

</br>

</br>

<code>

```
$("#container").ejChart({
  primaryXAxis: { desiredIntervals: 4 }
});
```

</code>

</td>

<td>

Property:<i>desiredIntervals</i>

</br>

</br>

<code>

```
let chart: Chart = new Chart({
```

```

primaryXAxis: { desiredIntervals: 4 }
});
chart.appendTo('#chart');
</code>
</td>
</tr>
<tr>
<td><b>Edges primaryXAxis</b></td>
<td>
<b>Property</b>:<i>edgeLabelPlacement</i>
</br>
</br>
<code>
$("#container").ejChart({
primaryXAxis: { edgeLabelPlacement: 'none' }
});
</code>
</td>
<td>
<b>Property</b>:<i>edgeLabelPlacement</i>
</br>
</br>
<code>
let chart: Chart = new Chart({
primaryXAxis: { edgeLabelPlacement: 'Shift' }
});
chart.appendTo('#chart');
</code>
</td>
</tr>
<tr>
<td><b>Enables trim for axis labels</b></td>
<td>

```

Property: *enableTrim*

```
$("#container").ejChart({  
  primaryXAxis: { enableTrim: true }  
});
```

Property: *enableTrim*

```
let chart: Chart = new Chart({  
  primaryXAxis: { enableTrim: true }  
});
```

```
chart.appendTo('#chart');
```

Specifies the interval of the axis according to the zoomed data of the chart

Property: *enableAutoIntervalOnZooming*

```
$("#container").ejChart({  
  primaryXAxis: { enableAutoIntervalOnZooming: true }  
});
```

<td>

Property:<i>enableAutoIntervalOnZooming</i>

</br>

</br>

<code>

```
let chart: Chart = new Chart({
  primaryXAxis: { enableAutoIntervalOnZooming: true }
});
```

```
chart.appendTo('#chart');
```

</code>

</td>

</tr>

<tr>

<td>Specifies the interval of the axis according to the zoomed data of the chart</td>

<td>

Property:<i>enableAutoIntervalOnZooming</i>

</br>

</br>

<code>

```
$("#container").ejChart({
  primaryXAxis: { enableAutoIntervalOnZooming: true }
});
```

</code>

</td>

<td>

Property:<i>enableAutoIntervalOnZooming</i>

</br>

</br>

<code>

```
let chart: Chart = new Chart({
  primaryXAxis: { enableAutoIntervalOnZooming: true }
});
```

```
chart.appendTo('#chart');
```

```
</code>
</td>
</tr>
<tr>
<td><b>Font style for primaryXAxis</b></td>
<td>
<b>Property</b>:<i>font</i>
</br>
</br>
<code>
$( "#container" ).ejChart({
primaryXAxis: { font: { fontFamily: 'Calibri', fontStyle: 'italic', fontWeight: '', opacity: 0.5, size: 12 } }
});
</code>
</td>
<td>
<b>Property</b>:<i>titleStyle</i>
</br>
</br>
<code>
let chart: Chart = new Chart({
primaryXAxis: { titleStyle: { } }
});
chart.appendTo('#chart');
</code>
</td>
</tr>
<tr>
<td><b>Indexed for category axis</b></td>
<td>
<b>Property</b>:<i>isIndexed</i>
</br>
</br>
</td>
```

```
<code>
```

```
$("#container").ejChart({  
  primaryXAxis: { isIndexed: true }  
});
```

```
</code>
```

```
</td>
```

```
<td>
```

```
<b>Property</b>:<i>isIndexed</i>
```

```
</br>
```

```
</br>
```

```
<code>
```

```
let chart: Chart = new Chart({  
  primaryXAxis: { isIndexed: true }  
});
```

```
chart.appendTo('#chart');
```

```
</code>
```

```
</td>
```

```
</tr>
```

```
<tr>
```

```
<td><b>Interval type for date time axis</b></td>
```

```
<td>
```

```
<b>Property</b>:<i>intervalType</i>
```

```
</br>
```

```
</br>
```

```
<code>
```

```
$("#container").ejChart({  
  primaryXAxis: { intervalType: 'Auto' }  
});
```

```
</code>
```

```
</td>
```

```
<td>
```

```
<b>Property</b>:<i>intervalType</i>
```

```
</br>
```

</br>

<code>

```
let chart: Chart = new Chart({
  primaryXAxis: { intervalType: 'Auto' }
});
```

```
chart.appendTo('#chart');
```

</code>

</td>

</tr>

<tr>

<td>Inversed axis</td>

<td>

Property:<i>isInversed</i>

</br>

</br>

<code>

```
$("#container").ejChart({
  primaryXAxis: { isInversed: true }
});
```

</code>

</td>

<td>

Property:<i>isInversed</i>

</br>

</br>

<code>

```
let chart: Chart = new Chart({
  primaryXAxis: { isInversed: true }
});
```

```
chart.appendTo('#chart');
```

</code>

</td>

</tr>

```

<tr>
<td><b>Custom label format</b></td>
<td>
<b>Property</b><i>labelFormat</i>
</br>
</br>
<code>
$( "#container" ).ejChart({
primaryXAxis: { labelFormat: '{value}K' }
});
</code>
</td>
<td>
<b>Property</b><i>labelFormat</i>
</br>
</br>
<code>
let chart: Chart = new Chart({
primaryXAxis: { labelFormat: '{value}K' }
});
chart.appendTo('#chart');
</code>
</td>
</tr>
<tr>
<td><b>labelIntersectAction</b></td>
<td>
<b>Property</b><i>labelIntersectAction</i>
</br>
</br>
<code>
$( "#container" ).ejChart({
primaryXAxis: { labelIntersectAction: 'trim' }

```



```

});
</code>
</td>
<td>
<b>Property</b>:<i>labelIntersectAction</i>
</br>
</br>
<code>
let chart: Chart = new Chart({
primaryXAxis: { labelIntersectAction: 'Trim' }
});
chart.appendTo('#chart');
</code>
</td>
</tr>
<tr>
<td><b>labelPosition</b></td>
<td>
<b>Property</b>:<i>labelPosition</i>
</br>
</br>
<code>
$("#container").ejChart({
primaryXAxis: { labelPosition: 'inside' }
});
</code>
</td>
<td>
<b>Property</b>:<i>labelPosition</i>
</br>
</br>
<code>
let chart: Chart = new Chart({

```

```

primaryXAxis: { labelPosition: 'Inside' }
});
chart.appendTo('#chart');
</code>
</td>
</tr>
<tr>
<td><b>labelPlacement for category axis</b></td>
<td>
<b>Property</b>:<i>labelPlacement</i>
</br>
</br>
<code>
$("#container").ejChart({
primaryXAxis: { labelPlacement: 'onTicks' }
});
</code>
</td>
<td>
<b>Property</b>:<i>labelPlacement</i>
</br>
</br>
<code>
let chart: Chart = new Chart({
primaryXAxis: { labelPlacement: 'OnTicks' }
});
chart.appendTo('#chart');
</code>
</td>
</tr>
<tr>
<td><b>Axis label alignment</b></td>
<td>

```

Property: *alignment*

```
$("#container").ejChart({  
  primaryXAxis: { alignment: 'center' }  
});
```

Not Applicable

Rotation of axis labels

Property: *labelRotation*

```
$("#container").ejChart({  
  primaryXAxis: { labelRotation: 45 }  
});
```

Property: *labelRotation*

```
let chart: Chart = new Chart({  
  primaryXAxis: { labelRotation: 45 }  
});
```

```

chart.appendTo('#chart');
</code>
</td>
</tr>
<tr>
<td><b>Log base value for logarithmic axis</b></td>
<td>
<b>Property</b>:<i>logBase</i>
</br>
</br>
<code>
$( "#container" ).ejChart({
primaryXAxis: { logBase: 10 }
});
</code>
</td>
<td>
<b>Property</b>:<i>labelRotation</i>
</br>
</br>
<code>
let chart: Chart = new Chart({
primaryXAxis: { logBase: 10 }
});
chart.appendTo('#chart');
</code>
</td>
</tr>
<tr>
<td><b>Major grid line</b></td>
<td>
<b>Property</b>:<i>majorGridLines.visible</i>
</br>

```

</br>

<code>

```
$("#container").ejChart({  
  primaryXAxis: { majorGridLines: { visible: true} }  
});
```

</code>

</td>

<td>

Not Applicable

</td>

</tr>

<tr>

<td>Width of MajorGridLines</td>

<td>

Property:<i>majorGridLines.width</i>

</br>

</br>

<code>

```
$("#container").ejChart({  
  primaryXAxis: { majorGridLines: { width: 2} }  
});
```

</code>

</td>

<td>

Property:<i>majorGridLines.width</i>

</br>

</br>

<code>

```
let chart: Chart = new Chart({  
  primaryXAxis: { majorGridLines: { width: 2} }  
});
```

```
chart.appendTo('#chart');
```

</code>

```
</td>
</tr>
<tr>
<td><b>Color of MajorGridLines</b></td>
<td>
<b>Property</b>:<i>majorGridLines.color</i>
</br>
</br>
<code>
$("#container").ejChart({
primaryXAxis: { majorGridLines: { color: 'black' } }
});
</code>
</td>
<td>
<b>Property</b>:<i>majorGridLines.color</i>
</br>
</br>
<code>
let chart: Chart = new Chart({
primaryXAxis: { majorGridLines: { color: 'black' } }
});
chart.appendTo('#chart');
</code>
</td>
</tr>
<tr>
<td><b>DashArray of MajorGridLines</b></td>
<td>
<b>Property</b>:<i>majorGridLines.dashArray</i>
</br>
</br>
<code>
```

```
$("#container").ejChart({
  primaryXAxis: { majorGridLines: { dashArray: 'black' } }
});
```

```
</code>
```

```
</td>
```

```
<td>
```

```
<b>Property</b>:<i>majorGridLines.dashArray</i>
```

```
</br>
```

```
</br>
```

```
<code>
```

```
let chart: Chart = new Chart({
  primaryXAxis: { majorGridLines: { dashArray: 'black' } }
});
```

```
chart.appendTo('#chart');
```

```
</code>
```

```
</td>
```

```
</tr>
```

```
<tr>
```

```
<td><b>Opacity of major grid line</b></td>
```

```
<td>
```

```
<b>Property</b>:<i>majorGridLines.opacity</i>
```

```
</br>
```

```
</br>
```

```
<code>
```

```
$("#container").ejChart({
  primaryXAxis: { majorGridLines: { opacity: true } }
});
```

```
</code>
```

```
</td>
```

```
<td>
```

```
Not Applicable
```

```
</td>
```

```
</tr>
```

<div><tr> <td>Major Tick line</td> <td> Property:<i>majorTickLines.visible</i> </br> </br> <code> \$("#container").ejChart({ primaryXAxis: { majorTickLines: { visible: true } } }); </code> </td> <td> Not Applicable </td> </tr> <tr> <td>Width of MajorTickLines</td> <td> Property:<i>majorTickLines.width</i> </br> </br> <code> \$("#container").ejChart({ primaryXAxis: { majorTickLines: { width: 2 } } }); </code> </td> <td> Property:<i>majorTickLines.width</i> </br> </br> <code></div>	
---	--


```
let chart: Chart = new Chart({
primaryXAxis: { majorTickLines: { width: 2} }
});
chart.appendTo('#chart');
```

```
</code>
```

```
</td>
```

```
</tr>
```

```
<tr>
```

```
<td><b>Height of MajorTickLines</b></td>
```

```
<td>
```

```
<b>Property</b>:<i>majorTickLines.size</i>
```

```
</br>
```

```
</br>
```

```
<code>
```

```
$("#container").ejChart({
primaryXAxis: { majorTickLines: { size: 2} }
});
```

```
</code>
```

```
</td>
```

```
<td>
```

```
<b>Property</b>:<i>majorTickLines.height</i>
```

```
</br>
```

```
</br>
```

```
<code>
```

```
let chart: Chart = new Chart({
primaryXAxis: { majorTickLines: { height: 2} }
});
chart.appendTo('#chart');
```

```
</code>
```

```
</td>
```

```
</tr>
```

```
<tr>
```

```
<td><b>Color of MajorTickLines</b></td>
```

<pre><td> Property:<i>majorTickLines.color</i> </br> </br> <code> \$("#container").ejChart({ primaryXAxis: { majorTickLines: { color: 'black' } } }); </code> </td> <td><pre><td> Property:<i>majorTickLines.color</i> </br> </br> <code> let chart: Chart = new Chart({ primaryXAxis: { majorTickLines: { color: 'black' } } }); chart.appendTo('#chart'); </code> </td> </pre></td></pre>	<pre><td> Property:<i>majorTickLines.color</i> </br> </br> <code> let chart: Chart = new Chart({ primaryXAxis: { majorTickLines: { color: 'black' } } }); chart.appendTo('#chart'); </code> </td> </pre>
<pre></tr> <tr> <td>Opacity of major Tick line</td> <td> Property:<i>majorTickLines.opacity</i> </br> </br> <code> \$("#container").ejChart({ primaryXAxis: { majorTickLines: { opacity: true } } }); </code> </pre>	

</td>

<td>

Not Applicable

</td>

</tr>

<tr>

<td>maximum labels of primaryXAxis</td>

<td>

Property:<i>maximumLabels</i>

</br>

</br>

<code>

```
$("#container").ejChart({  
primaryXAxis: { maximumLabels: 5 }  
});
```

</code>

</td>

<td>

Property:<i>maximumLabels</i>

</br>

</br>

<code>

```
let chart: Chart = new Chart({  
primaryXAxis: { maximumLabels: 4 }  
});  
chart.appendTo('#chart');
```

</code>

</td>

</tr>

<tr>

<td>maximum labels width of primaryXAxis to trim</td>

<td>

Property:<i>maximumLabelWidth</i>

</br>

</br>

<code>

```
$("#container").ejChart({
  primaryXAxis: { maximumLabelWidth: 40 }
});
```

</code>

</td>

<td>

Property:<i>maximumLabelWidth</i>

</br>

</br>

<code>

```
let chart: Chart = new Chart({
  primaryXAxis: { maximumLabelWidth: 4 }
});
```

```
chart.appendTo('#chart');
```

</code>

</td>

</tr>

<tr>

<td>minor grid line</td>

<td>

Property:<i>minorGridLines.visible</i>

</br>

</br>

<code>

```
$("#container").ejChart({
  primaryXAxis: { minorGridLines: { visible: true } }
});
```

</code>

</td>

<td>

Not Applicable

</td>

</tr>

<tr>

<td>Width of minorGridLines</td>

<td>

Property:<i>minorGridLines.width</i>

</br>

</br>

<code>

```
$("#container").ejChart({  
  primaryXAxis: { minorGridLines: { width: 2 } }  
});
```

</code>

</td>

<td>

Property:<i>minorGridLines.width</i>

</br>

</br>

<code>

```
let chart: Chart = new Chart({  
  primaryXAxis: { minorGridLines: { width: 2 } }  
});
```

```
chart.appendTo('#chart');
```

</code>

</td>

</tr>

<tr>

<td>Color of minorGridLines</td>

<td>

Property:<i>minorGridLines.color</i>

</br>

</br>

```
<code>
```

```
$("#container").ejChart({
  primaryXAxis: { minorGridLines: { color: 'black' } }
});
```

```
</code>
```

```
</td>
```

```
<td>
```

```
<b>Property</b>:<i>minorGridLines.color</i>
```

```
</br>
```

```
</br>
```

```
<code>
```

```
let chart: Chart = new Chart({
  primaryXAxis: { minorGridLines: { color: 'black' } }
});
chart.appendTo('#chart');
```

```
</code>
```

```
</td>
```

```
</tr>
```

```
<tr>
```

```
<td><b>DashArray of minorGridLines</b></td>
```

```
<td>
```

```
<b>Property</b>:<i>minorGridLines.dashArray</i>
```

```
</br>
```

```
</br>
```

```
<code>
```

```
$("#container").ejChart({
  primaryXAxis: { minorGridLines: { dashArray: 'black' } }
});
```

```
</code>
```

```
</td>
```

```
<td>
```

```
<b>Property</b>:<i>minorGridLines.dashArray</i>
```

```
</br>
```

</br>

<code>

```
let chart: Chart = new Chart({
primaryXAxis: { minorGridLines: { dashArray: 'black' } }
});
chart.appendTo('#chart');
```

</code>

</td>

</tr>

<tr>

<td>Opacity of minor grid line</td>

<td>

Property:<i>minorGridLines.opacity</i>

</br>

</br>

<code>

```
$("#container").ejChart({
primaryXAxis: { minorGridLines: { opacity: true } }
});
```

</code>

</td>

<td>

Not Applicable

</td>

</tr>

<tr>

<td>minor Tick line</td>

<td>

Property:<i>minorTickLines.visible</i>

</br>

</br>

<code>

```
$("#container").ejChart({
```

```
primaryXAxis: { minorTickLines: { visible: true } }
});
```

```
</code>
```

```
</td>
```

```
<td>
```

Not Applicable

```
</td>
```

```
</tr>
```

```
<tr>
```

```
<td><b>Width of minorTickLines</b></td>
```

```
<td>
```

```
<b>Property</b>:<i>minorTickLines.width</i>
```

```
</br>
```

```
</br>
```

```
<code>
```

```
$("#container").ejChart({
primaryXAxis: { minorTickLines: { width: 2 } }
});
```

```
</code>
```

```
</td>
```

```
<td>
```

```
<b>Property</b>:<i>minorTickLines.width</i>
```

```
</br>
```

```
</br>
```

```
<code>
```

```
let chart: Chart = new Chart({
primaryXAxis: { minorTickLines: { width: 2 } }
});
```

```
chart.appendTo('#chart');
```

```
</code>
```

```
</td>
```

```
</tr>
```

```
<tr>
```


<td>Height of minorTickLines</td>
--

<td>

Property:<i>minorTickLines.size</i>
--

</br>

</br>

<code>

<pre>\$("#container").ejChart({ primaryXAxis: { minorTickLines: { size: 2 } }};</pre>

</code>

</td>

<td>

Property:<i>minorTickLines.height</i>
--

</br>

</br>

<code>

<pre>let chart: Chart = new Chart({ primaryXAxis: { minorTickLines: { height: 2 } } }}; chart.appendTo('#chart');</pre>

</code>

</td>

</tr>

<tr>

<td>Color of minorTickLines</td>

<td>

Property:<i>minorTickLines.color</i>

</br>

</br>

<code>

<pre>\$("#container").ejChart({ primaryXAxis: { minorTickLines: { color: 'black' } } }};</pre>
--

</code>

</td>

<td>

Property:<i>minorTickLines.color</i>

</br>

</br>

<code>

```
let chart: Chart = new Chart({
  primaryXAxis: { minorTickLines: { color: 'black' } }
});
```

```
chart.appendTo('#chart');
```

</code>

</td>

</tr>

<tr>

<td>Opacity of minor Tick line</td>

<td>

Property:<i>minorTickLines.opacity</i>

</br>

</br>

<code>

```
$("#container").ejChart({
  primaryXAxis: { minorTickLines: { opacity: true } }
});
```

</code>

</td>

<td>

Not Applicable

</td>

</tr>

<tr>

<td>Minor ticks per interval of primaryXAxis</td>

<td>

Property: *minorTicksPerInterval*

`<`

```
$("#container").ejChart({
```

```
  primaryXAxis: { minorTicksPerInterval: 4 }
```

```
});
```

`>`

`<`

Property: *minorTickLines.color*

`<`

```
let chart: Chart = new Chart({
```

```
  primaryXAxis: { minorTicksPerInterval: 4 }
```

```
});
```

```
chart.appendTo('#chart');
```

`>`

`<`

`<`**name of the primaryXAxis**`>`

`<`

Property: *name*

`<`

```
$("#container").ejChart({
```

```
  primaryXAxis: { name: 'primaryXAxis' }
```

```
});
```

`>`

<pre> <td> Property:<i>name</i> </br> </br> <code> let chart: Chart = new Chart({ primaryXAxis: { name: 'primaryXAxis' } }); chart.appendTo('#chart'); </code> </td> </tr> <tr> <td> <td>Orientation of primaryXAxis</td> <td> Property:<i>orientation</i> </br> </br> <code> \$("#container").ejChart({ primaryXAxis: { orientation: 'Vertical' } }); </code> </td> <td> Not Applicable </td> </tr> <tr> <td> <td>Plot offset for primaryXAxis</td> <td> Property:<i>plotOffset</i> </br> </td></td></pre>	<td>Orientation of primaryXAxis</td> <td> Property:<i>orientation</i> </br> </br> <code> \$("#container").ejChart({ primaryXAxis: { orientation: 'Vertical' } }); </code> </td> <td> Not Applicable </td> </tr> <tr> <td> <td>Plot offset for primaryXAxis</td> <td> Property:<i>plotOffset</i> </br> </td>	<td>Plot offset for primaryXAxis</td> <td> Property:<i>plotOffset</i> </br>
--	---	---

</br>

<code>

```
$("#container").ejChart({  
  primaryXAxis: { plotOffset: 0 }  
});
```

</code>

</td>

<td>

Property:<i>plotOffset</i>

</br>

</br>

<code>

```
let chart: Chart = new Chart({  
  primaryXAxis: { plotOffset: 0 }  
});
```

```
chart.appendTo('#chart');
```

</code>

</td>

</tr>

<tr>

<td>minimum for primaryXAxis</td>

<td>

Property:<i>range.minimum</i>

</br>

</br>

<code>

```
$("#container").ejChart({  
  primaryXAxis: { range: { minimum: 10 } }  
});
```

</code>

</td>

<td>

Property:<i>minimum</i>

</br>

</br>

<code>

```
let chart: Chart = new Chart({
  primaryXAxis: { minimum: 23 }
});
```

```
chart.appendTo('#chart');
```

</code>

</td>

</tr>

<tr>

<td>maximum for primaryXAxis</td>

<td>

Property:<i>range.maximum</i>

</br>

</br>

<code>

```
$("#container").ejChart({
  primaryXAxis: { range: { maximum: 10 }}
});
```

</code>

</td>

<td>

Property:<i>maximum</i>

</br>

</br>

<code>

```
let chart: Chart = new Chart({
  primaryXAxis: { maximum: 23 }
});
```

```
chart.appendTo('#chart');
```

</code>

</td>

```
</tr>
<tr>
<td><b>interval for primaryXAxis</b></td>
<td>
<b>Property</b>:<i>range.interval</i>
</br>
</br>
<code>
$( "#container" ).ejChart({
primaryXAxis: { range: { interval: 1 }}
});
</code>
</td>
<td>
<b>Property</b>:<i>interval</i>
</br>
</br>
<code>
let chart: Chart = new Chart({
primaryXAxis: { interval: 2 }
});
chart.appendTo('#chart');
</code>
</td>
</tr>
<tr>
<td><b>RangePadding for primaryXAxis</b></td>
<td>
<b>Property</b>:<i>rangePadding</i>
</br>
</br>
<code>
$( "#container" ).ejChart({
```

```
primaryXAxis: { rangePadding: 'None' }}  
});
```

```
</code>
```

```
</td>
```

```
<td>
```

```
<b>Property</b>:<i>rangePadding</i>
```

```
</br>
```

```
</br>
```

```
<code>
```

```
let chart: Chart = new Chart({  
primaryXAxis: { rangePadding: 'None' }  
});
```

```
chart.appendTo('#chart');
```

```
</code>
```

```
</td>
```

```
</tr>
```

```
<tr>
```

```
<td><b>Rounding Places in primaryXAxis</b></td>
```

```
<td>
```

```
<b>Property</b>:<i>roundingPlaces </i>
```

```
</br>
```

```
</br>
```

```
<code>
```

```
$("#container").ejChart({  
primaryXAxis: { roundingPlaces: 3 }  
});
```

```
</code>
```

```
</td>
```

```
<td>
```

```
<b>Property</b>:<i>labelFormat</i>
```

```
</br>
```

```
</br>
```

```
<code>
```



```
let chart: Chart = new Chart({
primaryXAxis: { labelFormat: 'n3' }
});
chart.appendTo('#chart');
```

</code>

</td>

</tr>

<tr>

<td>ScrollBar settings of primaryXAxis</td>

<td>

Property:<i>scrollbarSettings </i>

</br>

</br>

<code>

```
$("#container").ejChart({
primaryXAxis: { scrollbarSettings : { } }
});
```

</code>

</td>

<td>

Not Applicable

</td>

</tr>

<tr>

<td>TickPosition in primaryXAxis</td>

<td>

Property:<i>tickLinesPosition</i>

</br>

</br>

<code>

```
$("#container").ejChart({
primaryXAxis: { tickLinesPosition: 'Inside' }
});
```

</code>

</td>

<td>

Property:<i>tickPosition</i>

</br>

</br>

<code>

```
let chart: Chart = new Chart({
  primaryXAxis: { tickPosition: 'Inside' }
});
```

```
chart.appendTo('#chart');
```

</code>

</td>

</tr>

<tr>

<td>valueType of primaryXAxis</td>

<td>

Property:<i>valueType</i>

</br>

</br>

<code>

```
$("#container").ejChart({
  primaryXAxis: { valueType: 'DateTime' }
});
```

</code>

</td>

<td>

Property:<i>valueType</i>

</br>

</br>

<code>

```
let chart: Chart = new Chart({
  primaryXAxis: { valueType: 'DateTime' }
```

```

});
chart.appendTo('#chart');
</code>
</td>
</tr>
<tr>
<td><b>visible of primaryXAxis</b></td>
<td>
<b>Property</b>:<i>visible</i>
</br>
</br>
<code>
$("#container").ejChart({
primaryXAxis: { visible: true }
});
</code>
</td>
<td>
<b>Property</b>:<i>visible</i>
</br>
</br>
<code>
let chart: Chart = new Chart({
primaryXAxis: { visible: true }
});
chart.appendTo('#chart');
</code>
</td>
</tr>
<tr>
<td><b>zoomFactor of primaryXAxis</b></td>
<td>
<b>Property</b>:<i>zoomFactor</i>

```

</br>

</br>

<code>

```
$("#container").ejChart({
  primaryXAxis: { zoomFactor: 0.3 }
});
```

</code>

</td>

<td>

Property:<i>zoomFactor</i>

</br>

</br>

<code>

```
let chart: Chart = new Chart({
  primaryXAxis: { zoomFactor: 0.3 }
});
```

```
chart.appendTo('#chart');
```

</code>

</td>

</tr>

<tr>

<td>zoomPosition of primaryXAxis</td>

<td>

Property:<i>zoomPosition</i>

</br>

</br>

<code>

```
$("#container").ejChart({
  primaryXAxis: { zoomPosition: 0.3 }
});
```

</code>

</td>

<td>

```

<b>Property</b>:<i>zoomPosition</i>
</br>
</br>
<code>
let chart: Chart = new Chart({
primaryXAxis: { zoomPosition: 0.3 }
});
chart.appendTo('#chart');
</code>
</td>
</tr>
<tr>
<td><b>labelBorder of primaryXAxis</b></td>
<td>
<b>Property</b>:<i>labelBorder</i>
</br>
</br>
<code>
$("#container").ejChart({
primaryXAxis: { labelBorder: { color: 'red', width: 2 } }
});
</code>
</td>
<td>
<b>Property</b>:<i>border</i>
</br>
</br>
<code>
let chart: Chart = new Chart({
primaryXAxis: { border: { color: 'red', width: 3 } }
});
chart.appendTo('#chart');
</code>

```

```
</td>
</tr>
<tr>
<td><b>title of primaryXAxis</b></td>
<td>
<b>Property</b>:<i>title.text</i>
</br>
</br>
<code>
$( "#container" ).ejChart({
primaryXAxis: { title: { text: 'Chart title' } }
});
</code>
</td>
<td>
<b>Property</b>:<i>title</i>
</br>
</br>
<code>
let chart: Chart = new Chart({
primaryXAxis: { title: 'Chart title' }
});
chart.appendTo('#chart');
</code>
</td>
</tr>
<tr>
<td><b>Stripline of primaryXAxis</b></td>
<td>
<b>Property</b>:<i>stripline</i>
</br>
</br>
<code>
```

```
$("#container").ejChart({
  primaryXAxis: { stripLine: [] }
});
```

```
</code>
```

```
</td>
```

```
<td>
```

```
<b>Property</b>:<i>stripLines</i>
```

```
</br>
```

```
</br>
```

```
<code>
```

```
let chart: Chart = new Chart({
  primaryXAxis: { stripLines: [] }
});
```

```
chart.appendTo('#chart');
```

```
</code>
```

```
</td>
```

```
</tr>
```

```
<tr>
```

```
<td><b>Multilevel labels of primaryXAxis</b></td>
```

```
<td>
```

```
<b>Property</b>:<i>multiLevelLabels</i>
```

```
</br>
```

```
</br>
```

```
<code>
```

```
$("#container").ejChart({
  primaryXAxis: { multiLevelLabels: [] }
});
```

```
</code>
```

```
</td>
```

```
<td>
```

```
<b>Property</b>:<i>multiLevelLabels</i>
```

```
</br>
```

```
</br>
```

```
<code>
```

```
let chart: Chart = new Chart({  
  primaryXAxis: { stripLines: [] }  
});
```

```
chart.appendTo('#chart');
```

```
</code>
```

```
</td>
```

```
</tr>
```

```
<tr>
```

```
<td><b>skeleton for an axes</b></td>
```

```
<td>
```

Not Applicable

```
</td>
```

```
<td>
```

```
<b>Property</b>:<i>skeleton</i>
```

```
</br>
```

```
</br>
```

```
<code>
```

```
let chart: Chart = new Chart({  
  axes: [ { skeleton: 'yMd' } ]  
});
```

```
chart.appendTo('#chart');
```

```
</code>
```

```
</td>
```

```
</tr>
```

```
<tr>
```

```
<td><b>skeleton type for an axes</b></td>
```

```
<td>
```

Not Applicable

```
</td>
```

```
<td>
```

```
<b>Property</b>:<i>skeletonType</i>
```

```
</br>
```


</br>

<code>

```
let chart: Chart = new Chart({
axes: [ { skeletonType: 'DateTime' } ]
});
```

```
chart.appendTo('#chart');
```

</code>

</td>

</tr>

## primaryYAxis		
Behaviour	API in Essential JS 1	API in Essential JS 2
Alternate grid band	Property:alternateGridBand \$("#container").ejChart({ primaryYAxis: { alternateGridBand: { even: { fill: 'red' } } } });	Not applicable
Axis line cross value	Property:crossesAt\$("#container").ejChart({ primaryYAxis: { crossesAt: 0 } });	Property:crossesAtlet chart: Chart = new Chart({ primaryYAxis: { crossesAt: 4 } }); chart.appendTo('#chart');
axis name with which the axis line has to be crossed	Property:crossesInAxis\$("#container").ejChart({ primaryYAxis: { crossesInAxis: '' } });	Property:crossesInAxislet chart: Chart = new Chart({ primaryYAxis: { crossesInAxis: '' } }); chart.appendTo('#chart');
axis elements placed with axis line	Property:showNextToAxisLine \$("#container").ejChart({ primaryYAxis: { showNextToAxisLine : true } });	Property:placeNextToAxisLinelet chart: Chart = new Chart({ primaryYAxis: { placeNextToAxisLine: '' } }); chart.appendTo('#chart');
axis line style	Property:axisLine.color\$("#container").ejChart({ primaryYAxis: { axisLine: { color : 'red' } } });	Property:lineStyle.colorlet chart: Chart = new Chart({ primaryYAxis: { lineStyle: { color: 'black' } } }); chart.appendTo('#chart');
axis line dashArray	Property:axisLine.color\$("#container").ejChart({ primaryYAxis: { axisLine: { dashArray : '10, 5' } } });	Property:lineStyle.dashArraylet chart: Chart = new Chart({ primaryYAxis: {

		<pre>lineStyle: { dashArray: '10, 5' } });chart.appendTo('#cha rt');</pre>
Offset for axis	<pre>Property:axisLine.offset\$("#container").ejChart ({ primaryYAxis: { axisLine: { offset : 10 } } });</pre>	<pre>Property:plotOffsetlet chart: Chart = new Chart({ primaryYAxis: { plotOffset: 10 });chart.appendTo('#cha rt');</pre>
Visible of an axis	<pre>Property:axisLine.offset\$("#container").ejChart ({ primaryYAxis: { axisLine: { visible : false } } });</pre>	<pre>Property:visiblelet chart: Chart = new Chart({ primaryYAxis: { visible: false });chart.appendTo('#cha rt');</pre>
Width of an axis	<pre>Property:axisLine.width\$("#container").ejChart ({ primaryYAxis: { axisLine: { width : 2 } } });</pre>	<pre>Property:lineStyle.widthlet chart: Chart = new Chart({ primaryYAxis: { lineStyle: { width: 3 } });chart.appendTo('#cha rt');</pre>
Column index of an axis	<pre>Property:columnIndex\$("#container").ejChart({ primaryYAxis: { columnIndex: 2 } });</pre>	<pre>Property:columnIndexlet chart: Chart = new Chart({ primaryYAxis: { columnIndex: 2 });chart.appendTo('#cha rt');</pre>
span of an axis to place horizontally or vertically	<pre>Property:columnSpan\$("#container").ejChart({ primaryYAxis: { columnIndex: 2 } });</pre>	<pre>Property:spanlet chart: Chart = new Chart({ primaryYAxis: { span: 2 });chart.appendTo('#cha rt');</pre>
Crosshair label of an axis	<pre>Property:crossHairLabel.visible\$("#container").e jChart({ primaryYAxis: { crossHairLabel: { visible: true } } });</pre>	<pre>Property:crossHairTooltip.enablelet chart: Chart = new Chart({ primaryYAxis: { crossHairTooltip: { enable: true } });chart.appendTo('#ch art');</pre>
Crosshair label color of an axis	Not applicable	<pre>Property:crossHairTooltip.filllet chart: Chart = new Chart({ primaryYAxis: { crossHairTooltip: { fill: 'red' } });chart.appendTo('#ch art');</pre>

Crosshair label text style	Not applicable	Property:crossHairTooltip.textStyle let chart: Chart = new Chart({ primaryYAxis: { crossHairTooltip: { textStyle: { } } } });chart.appendTo('#chart');
Desired interval count for primaryYAxis	Property:desiredIntervals \$("#container").ejChart({ primaryYAxis: { desiredIntervals: 4 } });	Property:desiredIntervals let chart: Chart = new Chart({ primaryYAxis: { desiredIntervals: 4 } });chart.appendTo('#chart');
Edges primaryYAxis	Property:edgeLabelPlacement \$("#container").ejChart({ primaryYAxis: { edgeLabelPlacement: 'none' } });	Property:edgeLabelPlacement let chart: Chart = new Chart({ primaryYAxis: { edgeLabelPlacement: 'Shift' } });chart.appendTo('#chart');
Enables trim for axis labels	Property:enableTrim \$("#container").ejChart({ primaryYAxis: { enableTrim: true } });	Property:enableTrim let chart: Chart = new Chart({ primaryYAxis: { enableTrim: true } });chart.appendTo('#chart');
Specifies the interval of the axis according to the zoomed data of the chart	Property:enableAutoIntervalOnZooming \$("#container").ejChart({ primaryYAxis: { enableAutoIntervalOnZooming: true } });	Property:enableAutoIntervalOnZooming let chart: Chart = new Chart({ primaryYAxis: { enableAutoIntervalOnZooming: true } });chart.appendTo('#chart');
Specifies the interval of the axis according to the zoomed data of the chart	Property:enableAutoIntervalOnZooming \$("#container").ejChart({ primaryYAxis: { enableAutoIntervalOnZooming: true } });	Property:enableAutoIntervalOnZooming let chart: Chart = new Chart({ primaryYAxis: { enableAutoIntervalOnZooming: true } });chart.appendTo('#chart');
Font style for primaryYAxis	Property:font \$("#container").ejChart({ primaryYAxis: { font: { fontFamily: 'Calibri', fontStyle: 'italic', fontWeight: '', opacity: 0.5, size: 12 } } });	Property:titleStyle let chart: Chart = new Chart({ primaryYAxis: { titleStyle: { } } });chart.appendTo('#chart');

Indexed for category axis	Property:isIndexed <code>\$("#container").ejChart({ primaryYAxis: { isIndexed: true }});</code>	Property:isIndexed <code>let chart: Chart = new Chart({ primaryYAxis: { isIndexed: true }});chart.appendTo('#chart');</code>
Interval type for date time axis	Property:intervalType <code>\$("#container").ejChart({ primaryYAxis: { intervalType: 'Auto' }});</code>	Property:intervalType <code>let chart: Chart = new Chart({ primaryYAxis: { intervalType: 'Auto' }});chart.appendTo('#chart');</code>
Inversed axis	Property:isInversed <code>\$("#container").ejChart({ primaryYAxis: { isInversed: true }});</code>	Property:isInversed <code>let chart: Chart = new Chart({ primaryYAxis: { isInversed: true }});chart.appendTo('#chart');</code>
Custom label format	Property:labelFormat <code>\$("#container").ejChart({ primaryYAxis: { labelFormat: '{value}K' }});</code>	Property:labelFormat <code>let chart: Chart = new Chart({ primaryYAxis: { labelFormat: '{value}K' }});chart.appendTo('#chart');</code>
labelIntersect Action	Property:labelIntersectAction <code>\$("#container").ejChart({ primaryYAxis: { labelIntersectAction: 'trim' }});</code>	Property:labelIntersectAction <code>let chart: Chart = new Chart({ primaryYAxis: { labelIntersectAction: 'Trim' }});chart.appendTo('#chart');</code>
labelPosition	Property:labelPosition <code>\$("#container").ejChart({ primaryYAxis: { labelPosition: 'inside' }});</code>	Property:labelPosition <code>let chart: Chart = new Chart({ primaryYAxis: { labelPosition: 'Inside' }});chart.appendTo('#chart');</code>
labelPlacement for category axis	Property:labelPlacement <code>\$("#container").ejChart({ primaryYAxis: { labelPlacement: 'onTicks' }});</code>	Property:labelPlacement <code>let chart: Chart = new Chart({ primaryYAxis: { labelPlacement: 'OnTicks' }});chart.appendTo('#chart');</code>
Axis label alignment	Property:alignment <code>\$("#container").ejChart({ primaryYAxis: { alignment: 'center' }});</code>	Not Applicable

Rotation of axis labels	Property:labelRotation\$("#container").ejChart({ primaryYAxis: { labelRotation: 45 } });	Property:labelRotationlet chart: Chart = new Chart({ primaryYAxis: { labelRotation: 45 } });chart.appendTo('#chart');
Log base value for logarithmic axis	Property:logBase\$("#container").ejChart({ primaryYAxis: { logBase: 10 } });	Property:labelRotationlet chart: Chart = new Chart({ primaryYAxis: { logBase: 10 } });chart.appendTo('#chart');
Major grid line	Property:majorGridLines.visible\$("#container").ejChart({ primaryYAxis: { majorGridLines: { visible: true } } });	Not Applicable
Width of MajorGridLines	Property:majorGridLines.width\$("#container").ejChart({ primaryYAxis: { majorGridLines: { width: 2 } } });	Property:majorGridLines.width1let chart: Chart = new Chart({ primaryYAxis: { majorGridLines: { width: 2 } } });chart.appendTo('#chart');
Color of MajorGridLines	Property:majorGridLines.color\$("#container").ejChart({ primaryYAxis: { majorGridLines: { color: 'black' } } });	Property:majorGridLines.colorlet chart: Chart = new Chart({ primaryYAxis: { majorGridLines: { color: 'black' } } });chart.appendTo('#chart');
DashArray of MajorGridLines	Property:majorGridLines.dashArray\$("#container").ejChart({ primaryYAxis: { majorGridLines: { dashArray: 'black' } } });	Property:majorGridLines.dashArraylet chart: Chart = new Chart({ primaryYAxis: { majorGridLines: { dashArray: 'black' } } });chart.appendTo('#chart');
Opacity of major grid line	Property:majorGridLines.opacity\$("#container").ejChart({ primaryYAxis: { majorGridLines: { opacity: true } } });	Not Applicable
Major Tick line	Property:majorTickLines.visible\$("#container").ejChart({ primaryYAxis: { majorTickLines: { visible: true } } });	Not Applicable
Width of MajorTickLines	Property:majorTickLines.width\$("#container").ejChart({ primaryYAxis: { majorTickLines: { width: 2 } } });	Property:majorTickLines.width1let chart: Chart = new Chart({ primaryYAxis: { majorTickLines: { width: 2 } } });

		});chart.appendTo('#chart');
Height of MajorTickLines	Property:majorTickLines.size\$("#container").ejChart({ primaryYAxis: { majorTickLines: { size: 2 } } });	Property:majorTickLines.height1 et chart: Chart = new Chart({ primaryYAxis: { majorTickLines: { height: 2} }});chart.appendTo('#chart');
Color of MajorTickLines	Property:majorTickLines.color\$("#container").ejChart({ primaryYAxis: { majorTickLines: { color: 'black' } } });	Property:majorTickLines.color1 et chart: Chart = new Chart({ primaryYAxis: { majorTickLines: { color: 'black' } }});chart.appendTo('#chart');
Opacity of major Tick line	Property:majorTickLines.opacity\$("#container").ejChart({ primaryYAxis: { majorTickLines: { opacity: true } } });	Not Applicable
maximum labels of primaryYAxis	Property:maximumLabels\$("#container").ejChart({ primaryYAxis: { maximumLabels: 5 } });	Property:maximumLabels1 et chart: Chart = new Chart({ primaryYAxis: { maximumLabels: 4 }});chart.appendTo('#chart');
maximum labels width of primaryYAxis to trim	Property:maximumLabelWidth\$("#container").ejChart({ primaryYAxis: { maximumLabelWidth: 40 } });	Property:maximumLabelWidth1 et chart: Chart = new Chart({ primaryYAxis: { maximumLabelWidth: 4 }});chart.appendTo('#chart');
minor grid line	Property:minorGridLines.visible\$("#container").ejChart({ primaryYAxis: { minorGridLines: { visible: true } } });	Not Applicable
Width of minorGridLines	Property:minorGridLines.width\$("#container").ejChart({ primaryYAxis: { minorGridLines: { width: 2 } } });	Property:minorGridLines.width1 et chart: Chart = new Chart({ primaryYAxis: { minorGridLines: { width: 2} }});chart.appendTo('#chart');
Color of minorGridLines	Property:minorGridLines.color\$("#container").ejChart({ primaryYAxis: { minorGridLines: { color: 'black' } } });	Property:minorGridLines.color1 et chart: Chart = new Chart({ primaryYAxis: { minorGridLines: { color: 'black' } }});chart.appendTo('#chart');

DashArray of minorGridLines	Property:minorGridLines.dashArray\$("#container").ejChart({ primaryYAxis: { minorGridLines: { dashArray: 'black' } } });	Property:minorGridLines.dashArraylet chart: Chart = new Chart({ primaryYAxis: { minorGridLines: { dashArray: 'black' } } });chart.appendTo('#chart');
Opacity of minor grid line	Property:minorGridLines.opacity\$("#container").ejChart({ primaryYAxis: { minorGridLines: { opacity: true } } });	Not Applicable
minor Tick line	Property:minorTickLines.visible\$("#container").ejChart({ primaryYAxis: { minorTickLines: { visible: true } } });	Not Applicable
Width of minorTickLines	Property:minorTickLines.width\$("#container").ejChart({ primaryYAxis: { minorTickLines: { width: 2 } } });	Property:minorTickLines.widthlet chart: Chart = new Chart({ primaryYAxis: { minorTickLines: { width: 2 } } });chart.appendTo('#chart');
Height of minorTickLines	Property:minorTickLines.size\$("#container").ejChart({ primaryYAxis: { minorTickLines: { size: 2 } } });	Property:minorTickLines.heightlet chart: Chart = new Chart({ primaryYAxis: { minorTickLines: { height: 2 } } });chart.appendTo('#chart');
Color of minorTickLines	Property:minorTickLines.color\$("#container").ejChart({ primaryYAxis: { minorTickLines: { color: 'black' } } });	Property:minorTickLines.colorlet chart: Chart = new Chart({ primaryYAxis: { minorTickLines: { color: 'black' } } });chart.appendTo('#chart');
Opacity of minor Tick line	Property:minorTickLines.opacity\$("#container").ejChart({ primaryYAxis: { minorTickLines: { opacity: true } } });	Not Applicable
Minor ticks per interval of primaryYAxis	Property:minorTicksPerInterval\$("#container").ejChart({ primaryYAxis: { minorTicksPerInterval: 4 } });	Property:minorTickLines.colorlet chart: Chart = new Chart({ primaryYAxis: { minorTicksPerInterval: 4 } });chart.appendTo('#chart');
name of the primaryYAxis	Property:name\$("#container").ejChart({ primaryYAxis: { name: 'primaryYAxis' } });	Property:namelet chart: Chart = new Chart({ primaryYAxis: { name: 'primaryYAxis' } });

		<code>}});chart.appendTo('#chart');</code>
Orientation of primaryYAxis	<code>Property:orientation\$("#container").ejChart({ primaryYAxis: { orientation: 'Vertical' } });</code>	Not Applicable
Plot offset for primaryYAxis	<code>Property:plotOffset\$("#container").ejChart({ primaryYAxis: { plotOffset: 0 } });</code>	<code>Property:plotOffsetlet chart: Chart = new Chart({ primaryYAxis: { plotOffset: 0 } });chart.appendTo('#chart');</code>
minimum for primaryYAxis	<code>Property:range.minimum\$("#container").ejChart({ primaryYAxis: { range: { minimum: 10 } } });</code>	<code>Property:minimumlet chart: Chart = new Chart({ primaryYAxis: { minimum: 23 } });chart.appendTo('#chart');</code>
maximum for primaryYAxis	<code>Property:range.maximum\$("#container").ejChart({ primaryYAxis: { range: { maximum: 10 } } });</code>	<code>Property:maximumlet chart: Chart = new Chart({ primaryYAxis: { maximum: 23 } });chart.appendTo('#chart');</code>
interval for primaryYAxis	<code>Property:range.interval\$("#container").ejChart({ primaryYAxis: { range: { interval: 1 } } });</code>	<code>Property:intervallet chart: Chart = new Chart({ primaryYAxis: { interval: 2 } });chart.appendTo('#chart');</code>
RangePadding for primaryYAxis	<code>Property:rangePadding\$("#container").ejChart({ primaryYAxis: { rangePadding: 'None' } });</code>	<code>Property:rangePaddinglet chart: Chart = new Chart({ primaryYAxis: { rangePadding: 'None' } });chart.appendTo('#chart');</code>
Rounding Places in primaryYAxis	<code>Property:roundingPlaces\$("#container").ejChart({ primaryYAxis: { roundingPlaces: 3 } });</code>	<code>Property:labelFormatlet chart: Chart = new Chart({ primaryYAxis: { labelFormat: 'n3' } });chart.appendTo('#chart');</code>
ScrollBar settings of primaryYAxis	<code>Property:scrollbarSettings\$("#container").ejChart({ primaryYAxis: { scrollbarSettings : { } } });</code>	Not Applicable
TickPosition in primaryYAxis	<code>Property:tickLinesPosition\$("#container").ejChart({ primaryYAxis: { tickLinesPosition: 'Inside' } });</code>	<code>Property:tickPositionlet chart: Chart = new Chart({ primaryYAxis: { tickPosition: 'Inside' }</code>

		<code>}});chart.appendTo('#chart');</code>
valueType of primaryYAxis	<code>Property:valueType\$("#container").ejChart({ primaryYAxis: { valueType: 'DateTime' } });</code>	Property:valueType <code>let chart: Chart = new Chart({ primaryYAxis: { valueType: 'DateTime' } });chart.appendTo('#chart');</code>
visible of primaryYAxis	<code>Property:visible\$("#container").ejChart({ primaryYAxis: { visible: true } });</code>	Property:visible <code>let chart: Chart = new Chart({ primaryYAxis: { visible: true } });chart.appendTo('#chart');</code>
zoomFactor of primaryYAxis	<code>Property:zoomFactor\$("#container").ejChart({ primaryYAxis: { zoomFactor: 0.3 } });</code>	Property:zoomFactor <code>let chart: Chart = new Chart({ primaryYAxis: { zoomFactor: 0.3 } });chart.appendTo('#chart');</code>
zoomPosition of primaryYAxis	<code>Property:zoomPosition\$("#container").ejChart({ primaryYAxis: { zoomPosition: 0.3 } });</code>	Property:zoomPosition <code>let chart: Chart = new Chart({ primaryYAxis: { zoomPosition: 0.3 } });chart.appendTo('#chart');</code>
labelBorder of primaryYAxis	<code>Property:labelBorder\$("#container").ejChart({ primaryYAxis: { labelBorder: { color: 'red', width: 2 } } });</code>	Property:border <code>let chart: Chart = new Chart({ primaryYAxis: { border: { color: 'red', width: 3 } } });chart.appendTo('#chart');</code>
title of primaryYAxis	<code>Property:title.text\$("#container").ejChart({ primaryYAxis: { title: { text: 'Chart title' } } });</code>	Property:title <code>let chart: Chart = new Chart({ primaryYAxis: { title: 'Chart title' } });chart.appendTo('#chart');</code>
StripLine of primaryYAxis	<code>Property:stripLine\$("#container").ejChart({ primaryYAxis: { stripLine: [] } });</code>	Property:stripLines <code>let chart: Chart = new Chart({ primaryYAxis: { stripLines: [] } });chart.appendTo('#chart');</code>
Multilevel labels of primaryYAxis	<code>Property:multiLevelLabels\$("#container").ejChart({ primaryYAxis: { multiLevelLabels: [] } });</code>	Property:multiLevelLabels <code>let chart: Chart = new Chart({ primaryYAxis: { stripLines: [] } });</code>

		<code>}});chart.appendTo('#chart');</code>
skeleton for an axes	Not Applicable	Property:skeleton <code>let chart: Chart = new Chart({ axes: [{ skeleton: 'yMd' }]});chart.appendTo('#chart');</code>
skeleton type for an axes	Not Applicable	Property:skeletonType <code>let chart: Chart = new Chart({ axes: [{ skeletonType: 'DateTime' }]});chart.appendTo('#chart');</code>
## Axes		
Behaviour	API in Essential JS 1	API in Essential JS 2
Alternate grid band	Property:alternateGridBand <code>\$("#container").ejChart({ axes: [{ alternateGridBand: { even: { fill: 'red' } } }]});</code>	Not applicable
Axis line cross value	Property:crossesAt <code>\$("#container").ejChart({ axes: [{ crossesAt: 0 }]});</code>	Property:crossesAt <code>let chart: Chart = new Chart({ axes: [{ crossesAt: 4 }]});chart.appendTo('#chart');</code>
axis name with which the axis line has to be crossed	Property:crossesInAxis <code>\$("#container").ejChart({ axes: [{ crossesInAxis: '' }]});</code>	Property:crossesInAxis <code>let chart: Chart = new Chart({ axes: [{ crossesInAxis: '' }]});chart.appendTo('#chart');</code>
axis elements placed with axis line	Property:showNextToAxisLine <code>\$("#container").ejChart({ axes: [{ showNextToAxisLine: true }]});</code>	Property:placeNextToAxisLine <code>let chart: Chart = new Chart({ axes: [{ placeNextToAxisLine: '' }]});chart.appendTo('#chart');</code>

axis line style	Property:axisLine.color\$(" #container").ejChart (({ axes: [{ axisLine: { color : 'red' } }]});	Property:lineStyle.colorlet chart: Chart = new Chart({ axes: [{ lineStyle: { color: 'black' } }]});chart.appendTo('#chart');
axis line dashAr ray	Property:axisLine.color\$(" #container").ejChart (({ axes: [{ axisLine: { dashArray : '10, 5' } }]});	Property:lineStyle.dashArraylet chart: Chart = new Chart({ axes: [{ lineStyle: { dashArray: '10, 5' } }]});chart.appendTo('#chart');
Offset for axis	Property:axisLine.offset\$(" #container").ejChart (({ axes: [{ axisLine: { offset : 10 } }]});	Property:plotOffsetlet chart: Chart = new Chart({ axes: [{ plotOffset: 10 }]});chart.appendTo('#chart');
Visible of an axis	Property:axisLine.offset\$(" #container").ejChart (({ axes: [{ axisLine: { visible : false } }]});	Property:visiblelet chart: Chart = new Chart({ axes: [{ visible: false }]});chart.appendTo('#chart');
Width of an axis	Property:axisLine.width\$(" #container").ejChart (({ axes: [{ axisLine: { width : 2 } }]});	Property:lineStyle.widthlet chart: Chart = new Chart({ axes: [{ lineStyle: { width: 3 } }]});chart.appendTo('#chart');
Colum n index of an axis	Property:columnIndex\$(" #container").ejChart({ axes: [{ columnIndex: 2 }]});	Property:columnIndexlet chart: Chart = new Chart({ axes: [{ columnIndex: 2 }]});chart.appendTo('#chart');
span of an axis to place horizon tally or vertical ly	Property:columnSpan\$(" #container").ejChart({ axes: [{ columnIndex: 2 }]});	Property:spanlet chart: Chart = new Chart({ axes: [{ span: 2 }]});chart.appendTo('#chart');
Crossh air label of an axis	Property:crossHairLabel.visi ble\$(" #container").ej Chart({ axes: [{ crossHairLabel: { visible: true } }]});	Property:crossHairTooltip.enablelet chart: Chart = new Chart({ axes: [{ crossHairTooltip: { enable: true } }]});chart.appendTo('#chart');
Crossh air label	Not applicable	Property:crossHairTooltip.filllet chart: Chart = new Chart({ axes: [{ crossHairTooltip: { fill: 'red' } }]});chart.appendTo('#chart');

color of an axis		
Crosshair label text style	Not applicable	Property:crossHairTooltip.textStyle let chart: Chart = new Chart({ axes: [{ crossHairTooltip: { textStyle: { } } }] });chart.appendTo('#chart');
Desired interval count for primary YAxis	Property:desiredIntervals\$ ("#container").ejChart({ axes: [{ desiredIntervals: 4 }] });	Property:desiredIntervals let chart: Chart = new Chart({ axes: [{ desiredIntervals: 4 }] });chart.appendTo('#chart');
Edges primary YAxis	Property:edgeLabelPlacement\$ ("#container").ejChart({ axes: [{ edgeLabelPlacement: 'none' }] });	Property:edgeLabelPlacement let chart: Chart = new Chart({ axes: [{ edgeLabelPlacement: 'Shift' }] });chart.appendTo('#chart');
Enable trim for axis labels	Property:enableTrim\$ ("#container").ejChart({ axes: [{ enableTrim: true }] });	Property:enableTrim let chart: Chart = new Chart({ axes: [{ enableTrim: true }] });chart.appendTo('#chart');
Specifies the interval of the axis according to the zoomed data of the chart	Property:enableAutoIntervalOnZooming\$ ("#container").ejChart({ axes: [{ enableAutoIntervalOnZooming: true }] });	Property:enableAutoIntervalOnZooming let chart: Chart = new Chart({ axes: [{ enableAutoIntervalOnZooming: true }] });chart.appendTo('#chart');
Specifies the interval of the axis according to the zoom	Property:enableAutoIntervalOnZooming\$ ("#container").ejChart({ axes: [{ enableAutoIntervalOnZooming: true }] });	Property:enableAutoIntervalOnZooming let chart: Chart = new Chart({ axes: [{ enableAutoIntervalOnZooming: true }] });chart.appendTo('#chart');

d data of the chart		
Font style for primary YAxis	Property:font\$("#container").ejChart({ axes: [{ font: { fontFamily: 'Calibri', fontStyle: 'italic', fontWeight: '', opacity: 0.5, size: 12} }] });	Property:titleStylelet chart: Chart = new Chart({ axes: [{ titleStyle: { } }] });chart.appendTo('#chart');
Indexed for category axis	Property:isIndexed\$("#container").ejChart({ axes: [{ isIndexed: true }] });	Property:isIndexedlet chart: Chart = new Chart({ axes: [{ isIndexed: true }] });chart.appendTo('#chart');
Interval type for date time axis	Property:intervalType\$("#container").ejChart({ axes: [{ intervalType: 'Auto' }] });	Property:intervalTypelet chart: Chart = new Chart({ axes: [{ intervalType: 'Auto' }] });chart.appendTo('#chart');
Inversed axis	Property:isInversed\$("#container").ejChart({ axes: [{ isInversed: true }] });	Property:isInversedlet chart: Chart = new Chart({ axes: [{ isInversed: true }] });chart.appendTo('#chart');
Custom label format	Property:labelFormat\$("#container").ejChart({ axes: [{ labelFormat: '{value}K' }] });	Property:labelFormatlet chart: Chart = new Chart({ axes: [{ labelFormat: '{value}K' }] });chart.appendTo('#chart');
labelIntersectAction	Property:labelIntersectAction\$("#container").ejChart({ axes: [{ labelIntersectAction: 'trim' }] });	Property:labelIntersectActionlet chart: Chart = new Chart({ axes: [{ labelIntersectAction: 'Trim' }] });chart.appendTo('#chart');
labelPosition	Property:labelPosition\$("#container").ejChart({ axes: [{ labelPosition: 'inside' }] });	Property:labelPositionlet chart: Chart = new Chart({ axes: [{ labelPosition: 'Inside' }] });chart.appendTo('#chart');
labelPlacement for	Property:labelPlacement\$("#container").ejChart({ axes: [{	Property:labelPlacementlet chart: Chart = new Chart({ axes: [{ labelPlacement: 'OnTicks' }] });chart.appendTo('#chart');

category axis	labelPlacement: 'onTicks' }]]]);	
Axis label alignment	Property:alignment\$("#container").ejChart({ axes: [{ alignment: 'center' }]]);	Not Applicable
Rotation of axis labels	Property:labelRotation\$("#container").ejChart({ axes: [{ labelRotation: 45 }]]);	Property:labelRotationlet chart: Chart = new Chart({ axes: [{ labelRotation: 45 }]]);chart.appendTo('#chart');
Log base value for logarithmic axis	Property:logBase\$("#container").ejChart({ axes: [{ logBase: 10 }]]);	Property:labelRotationlet chart: Chart = new Chart({ axes: [{ logBase: 10 }]]);chart.appendTo('#chart');
Major grid line	Property:majorGridLines.visible\$("#container").ejChart({ axes: [{ majorGridLines: { visible: true } }]]);	Not Applicable
Width of MajorGridLines	Property:majorGridLines.width\$("#container").ejChart({ axes: [{ majorGridLines: { width: 2 } }]]);	Property:majorGridLines.widthlet chart: Chart = new Chart({ axes: [{ majorGridLines: { width: 2 } }]]);chart.appendTo('#chart');
Color of MajorGridLines	Property:majorGridLines.color\$("#container").ejChart({ axes: [{ majorGridLines: { color: 'black' } }]]);	Property:majorGridLines.colorlet chart: Chart = new Chart({ axes: [{ majorGridLines: { color: 'black' } }]]);chart.appendTo('#chart');
DashArray of MajorGridLines	Property:majorGridLines.dashArray\$("#container").ejChart({ axes: [{ majorGridLines: { dashArray: 'black' } }]]);	Property:majorGridLines.dashArraylet chart: Chart = new Chart({ axes: [{ majorGridLines: { dashArray: 'black' } }]]);chart.appendTo('#chart');
Opacity of major	Property:majorGridLines.opacity\$("#container").ejChart({ axes: [{ majorGridLines: { opacity: true } }]]);	Not Applicable

grid line		
Major Tick line	Property:majorTickLines.visible\$("#container").ejChart({ axes: [{ majorTickLines: { visible: true} }]});	Not Applicable
Width of Major Tick Lines	Property:majorTickLines.width\$("#container").ejChart({ axes: [{ majorTickLines: { width: 2} }]});	Property:majorTickLines.widthlet chart: Chart = new Chart({ axes: [{ majorTickLines: { width: 2} }]});chart.appendTo('#chart');
Height of Major Tick Lines	Property:majorTickLines.size\$("#container").ejChart({ axes: [{ majorTickLines: { size: 2} }]});	Property:majorTickLines.heightlet chart: Chart = new Chart({ axes: [{ majorTickLines: { height: 2} }]});chart.appendTo('#chart');
Color of Major Tick Lines	Property:majorTickLines.color\$("#container").ejChart({ axes: [{ majorTickLines: { color: 'black' } }]});	Property:majorTickLines.colorlet chart: Chart = new Chart({ axes: [{ majorTickLines: { color: 'black' } }]});chart.appendTo('#chart');
Opacity of major Tick line	Property:majorTickLines.opacity\$("#container").ejChart({ axes: [{ majorTickLines: { opacity: true} }]});	Not Applicable
maximum labels of primary YAxis	Property:maximumLabels\$("#container").ejChart({ axes: [{ maximumLabels: 5 }]});	Property:maximumLabelslet chart: Chart = new Chart({ axes: [{ maximumLabels: 4 }]});chart.appendTo('#chart');
maximum labels width of primary YAxis to trim	Property:maximumLabelWidth\$("#container").ejChart({ axes: [{ maximumLabelWidth: 40 }]});	Property:maximumLabelWidthlet chart: Chart = new Chart({ axes: [{ maximumLabelWidth: 4 }]});chart.appendTo('#chart');

minor grid line	Property:minorGridLines.visible\$("#container").ejChart({ axes: [{ minorGridLines: { visible: true} }]});	Not Applicable
Width of minorGridLines	Property:minorGridLines.width\$("#container").ejChart({ axes: [{ minorGridLines: { width: 2} }]});	Property:minorGridLines.widthlet chart: Chart = new Chart({ axes: [{ minorGridLines: { width: 2} }]});chart.appendTo('#chart');
Color of minorGridLines	Property:minorGridLines.color\$("#container").ejChart({ axes: [{ minorGridLines: { color: 'black' } }]});	Property:minorGridLines.colorlet chart: Chart = new Chart({ axes: [{ minorGridLines: { color: 'black' } }]});chart.appendTo('#chart');
DashArray of minorGridLines	Property:minorGridLines.dashArray\$("#container").ejChart({ axes: [{ minorGridLines: { dashArray: 'black' } }]});	Property:minorGridLines.dashArraylet chart: Chart = new Chart({ axes: [{ minorGridLines: { dashArray: 'black' } }]});chart.appendTo('#chart');
Opacity of minor grid line	Property:minorGridLines.opacity\$("#container").ejChart({ axes: [{ minorGridLines: { opacity: true} }]});	Not Applicable
minor Tick line	Property:minorTickLines.visible\$("#container").ejChart({ axes: [{ minorTickLines: { visible: true} }]});	Not Applicable
Width of minorTickLines	Property:minorTickLines.width\$("#container").ejChart({ axes: [{ minorTickLines: { width: 2} }]});	Property:minorTickLines.widthlet chart: Chart = new Chart({ axes: [{ minorTickLines: { width: 2} }]});chart.appendTo('#chart');
Height of minorTickLines	Property:minorTickLines.size\$("#container").ejChart({ axes: [{ minorTickLines: { size: 2} }]});	Property:minorTickLines.heightlet chart: Chart = new Chart({ axes: [{ minorTickLines: { height: 2} }]});chart.appendTo('#chart');

Color of minor Tick Lines	Property:minorTickLines.color\$("#container").ejChart({ axes: [{ minorTickLines: { color: 'black' } }]});	Property:minorTickLines.colorlet chart: Chart = new Chart({ axes: [{ minorTickLines: { color: 'black' } }]});chart.appendTo('#chart');
Opacity of minor Tick line	Property:minorTickLines.opacity\$("#container").ejChart({ axes: [{ minorTickLines: { opacity: true } }]});	Not Applicable
Minor ticks per interval of primary YAxis	Property:minorTicksPerInterval\$("#container").ejChart({ axes: [{ minorTicksPerInterval: 4 } }]});	Property:minorTickLines.colorlet chart: Chart = new Chart({ axes: [{ minorTicksPerInterval: 4 } }]});chart.appendTo('#chart');
name of the primary YAxis	Property:name\$("#container").ejChart({ axes: [{ name: 'primaryYAxis' } }]});	Property:namelet chart: Chart = new Chart({ axes: [{ name: 'primaryYAxis' } }]});chart.appendTo('#chart');
Orientation of primary YAxis	Property:orientation\$("#container").ejChart({ axes: [{ orientation: 'Vertical' } }]});	Not Applicable
Plot offset for primary YAxis	Property:plotOffset\$("#container").ejChart({ axes: [{ plotOffset: 0 } }]});	Property:plotOffsetlet chart: Chart = new Chart({ axes: [{ plotOffset: 0 } }]});chart.appendTo('#chart');
minimum for primary YAxis	Property:range.minimum\$("#container").ejChart({ axes: [{ range: { minimum: 10 } }]});	Property:minimumlet chart: Chart = new Chart({ axes: [{ minimum: 23 } }]});chart.appendTo('#chart');
maximum for primary YAxis	Property:range.maximum\$("#container").ejChart({ axes: [{ range: { maximum: 10 } }]});	Property:maximumlet chart: Chart = new Chart({ axes: [{ maximum: 23 } }]});chart.appendTo('#chart');
interval for	Property:range.interval\$("#container").ejChart	Property:intervallet chart: Chart = new Chart({ axes: [{ interval: 2 } }]});chart.appendTo('#chart');

primaryYAxis	<pre>{ axes: [{ range: { interval: 1 } }] };</pre>	
RangePadding for primaryYAxis	<pre>Property:rangePadding\$("#container").ejChart ({ axes: [{ rangePadding: 'None' }] });</pre>	<pre>Property:rangePaddinglet chart: Chart = new Chart({ axes: [{ rangePadding: 'None' }] });chart.appendTo('#chart');</pre>
Rounding Places in primaryYAxis	<pre>Property:roundingPlaces \$("#container").ejChart ({ axes: [{ roundingPlaces: 3 }] });</pre>	<pre>Property:labelFormatlet chart: Chart = new Chart({ axes: [{ labelFormat: 'n3' }] });chart.appendTo('#chart');</pre>
Scrollbar settings of primaryYAxis	<pre>Property:scrollbarSettings \$("#container").ejChart ({ axes: [{ scrollbarSettings : { } }] });</pre>	Not Applicable
TickPosition in primaryYAxis	<pre>Property:tickLinesPosition\$("#container").ejChart ({ axes: [{ tickLinesPosition: 'Inside' }] });</pre>	<pre>Property:tickPositionlet chart: Chart = new Chart({ axes: [{ tickPosition: 'Inside' }] });chart.appendTo('#chart');</pre>
valueType of primaryYAxis	<pre>Property:valueType\$("#container").ejChart ({ axes: [{ valueType: 'DateTime' }] });</pre>	<pre>Property:valueTypelet chart: Chart = new Chart({ axes: [{ valueType: 'DateTime' }] });chart.appendTo('#chart');</pre>
visible of primaryYAxis	<pre>Property:visible\$("#container").ejChart ({ axes: [{ visible: true }] });</pre>	<pre>Property:visiblelet chart: Chart = new Chart({ axes: [{ visible: true }] });chart.appendTo('#chart');</pre>
zoomFactor of primaryYAxis	<pre>Property:zoomFactor\$("#container").ejChart ({ axes: [{ zoomFactor: 0.3 }] });</pre>	<pre>Property:zoomFactorlet chart: Chart = new Chart({ axes: [{ zoomFactor: 0.3 }] });chart.appendTo('#chart');</pre>
zoomPosition of primaryYAxis	<pre>Property:zoomPosition\$("#container").ejChart ({ axes: [{ zoomPosition: 0.3 }] });</pre>	<pre>Property:zoomPositionlet chart: Chart = new Chart({ axes: [{ zoomPosition: 0.3 }] });chart.appendTo('#chart');</pre>

labelBorder of primaryYAxis	Property:labelBorder\$("#container").ejChart({ axes: [{ labelBorder: { color: 'red', width: 2 } }] });	Property:borderlet chart: Chart = new Chart({ axes: [{ border: { color: 'red', width: 3 } }] });chart.appendTo('#chart');												
title of primaryYAxis	Property:title.text\$("#container").ejChart({ axes: [{ title: { text: 'Chart title' } }] });	Property:titlelet chart: Chart = new Chart({ axes: [{ title: 'Chart title' }] });chart.appendTo('#chart');												
StripLine of primaryYAxis	Property:stripline\$("#container").ejChart({ axes: [{ stripLine: [] }] });	Property:stripLineslet chart: Chart = new Chart({ axes: [{ stripLines: [] }] });chart.appendTo('#chart');												
Multilevel labels of axes	Property:multiLevelLabels\$("#container").ejChart({ axes: [{ multiLevelLabels: [] }] });	Property:multiLevelLabelslet chart: Chart = new Chart({ axes: [{ stripLines: [] }] });chart.appendTo('#chart');												
skeleton for an axes	Not Applicable	Property:skeletonlet chart: Chart = new Chart({ axes: [{ skeleton: 'yMd' }] });chart.appendTo('#chart');												
skeleton type for an axes	Not Applicable	Property:skeletonTypelet chart: Chart = new Chart({ axes: [{ skeletonType: 'DateTime' }] });chart.appendTo('#chart'); ## Rows <table> <tr> <td>Behavior</td><td>API in Essential JS 1</td><td>API in Essential JS 2</td></tr> <tr> <td>rows in chart</td><td>Property:rowDefinitions\$("#chart").ejChart({ rowDefinitions: [] });</td><td>Property:rowslet chart: Chart = new Chart({ rows: [] });chart.appendTo('#chart');</td></tr> <tr> <td>unit</td><td>Property:unit\$("#container").ejChart({ rowDefinitions: [{ unit: "percentage" }] });</td><td>Not Applicable</td></tr> <tr> <td>height of rows in chart</td><td>Property:rowHeight\$("#chart").ejChart({ rowDefinitions: [{ rowHeight: '50%' }] });</td><td>Property:heightlet chart: Chart = new Chart({ rows: [{ height: '300%' }] });chart.appendTo('#chart');</td></tr> </table>	Behavior	API in Essential JS 1	API in Essential JS 2	rows in chart	Property:rowDefinitions\$("#chart").ejChart({ rowDefinitions: [] });	Property:rowslet chart: Chart = new Chart({ rows: [] });chart.appendTo('#chart');	unit	Property:unit\$("#container").ejChart({ rowDefinitions: [{ unit: "percentage" }] });	Not Applicable	height of rows in chart	Property:rowHeight\$("#chart").ejChart({ rowDefinitions: [{ rowHeight: '50%' }] });	Property:heightlet chart: Chart = new Chart({ rows: [{ height: '300%' }] });chart.appendTo('#chart');
Behavior	API in Essential JS 1	API in Essential JS 2												
rows in chart	Property:rowDefinitions\$("#chart").ejChart({ rowDefinitions: [] });	Property:rowslet chart: Chart = new Chart({ rows: [] });chart.appendTo('#chart');												
unit	Property:unit\$("#container").ejChart({ rowDefinitions: [{ unit: "percentage" }] });	Not Applicable												
height of rows in chart	Property:rowHeight\$("#chart").ejChart({ rowDefinitions: [{ rowHeight: '50%' }] });	Property:heightlet chart: Chart = new Chart({ rows: [{ height: '300%' }] });chart.appendTo('#chart');												

		Property:lineColor, lineWidth\$("#chart").ej Line chart({ custom rowDefinitions: [{ ization rowHeight: '50%', lineColor: 'brown', lineWidth: 2}]);	Property:borderlet chart: Chart = new Chart({ rows: [{ height: '300', border: { width: 2, color: 'brown'}]); chart. appendTo('#chart');
		## Series	
	Behavior	API in Essential JS 1	API in Essential JS 2
	bearFillColor	Property:bearFillColor\$("#char t").ejChart({ series: [[bearFillColor: 'red']];});	Property:bearFillColor let chart: Chart = new Chart({ series: [[bearFillColor: 'red']];}); chart.appe ndTo('#chart');
	Border	Property:border\$("#chart"). ejChart({ series: [{ border: { color: 'red', width: 2, dashArray: '10, 5' } }]);	Property:rowslet chart: Chart = new Chart({ series: [{ border: { color: 'red', width: 2}]];}); chart.appe ndTo('#chart');
	BoxPlot Mode	Property:boxPlotMode\$("#cha rt").ejChart({ series: [[boxPlotMode: 'inclusive' }]);	Property:rowslet chart: Chart = new Chart({ series: [{ boxPlotMode: 'Inclusive']];}); chart.appe ndTo('#chart');
	Minimum radius of Bubble series	Property:bubbleOptions.minRad ius\$("#chart").ejChart({ series: [{ bubbleOptions: { minRadius: 2 } }]);	Property:minRadius1 et chart: Chart = new Chart({ series: [{ minRadius: 2]];}); chart.appe ndTo('#chart');
	Maximum radius of Bubble series	Property:bubbleOptions.maxRad ius\$("#chart").ejChart({ series: [{ bubbleOptions: { maxRadius: 10 } }]);	Property:maxRadius1 et chart: Chart = new Chart({ series: [{ maxRadius: 2]];});

		<pre> });});chart.appendTo('#chart'); Property:bullFillColor let chart: Chart = new Chart({ series: [{bullFillColor: 'red' }]);});chart.appendTo('#chart'); Property:cardinalSplineTension let chart: Chart = new Chart({ series: [{ cardinalSplineTension: 0.5 }]);});chart.appendTo('#chart'); Property:columnWidth let chart: Chart = new Chart({ series: [{ columnWidth: 0.5 }]);});chart.appendTo('#chart'); Property:columnSpacing let chart: Chart = new Chart({ series: [{ columnSpacing: 0.5 }]);});chart.appendTo('#chart'); Property:cornerRadius.topLeft let chart: Chart = new Chart({ series: [{ topLeft: 0 }]);});chart.appendTo('#chart'); Property:cornerRadius.topRight let chart: Chart = new Chart({ series: [{ </pre>
	<pre> });});chart.appendTo('#chart'); Property:bullFillColor let chart: Chart = new Chart({ series: [{bullFillColor: 'red' }]);});chart.appendTo('#chart'); Property:cardinalSplineTension let chart: Chart = new Chart({ series: [{ cardinalSplineTension: 0.5 }]);});chart.appendTo('#chart'); Property:columnWidth let chart: Chart = new Chart({ series: [{ columnWidth: 0.5 }]);});chart.appendTo('#chart'); Property:columnSpacing let chart: Chart = new Chart({ series: [{ columnSpacing: 0.5 }]);});chart.appendTo('#chart'); Property:cornerRadius.topLeft let chart: Chart = new Chart({ series: [{ topLeft: 0 }]);});chart.appendTo('#chart'); Property:cornerRadius.topRight let chart: Chart = new Chart({ series: [{ </pre>	<pre> });});chart.appendTo('#chart'); Property:bullFillColor let chart: Chart = new Chart({ series: [{bullFillColor: 'red' }]);});chart.appendTo('#chart'); Property:cardinalSplineTension let chart: Chart = new Chart({ series: [{ cardinalSplineTension: 0.5 }]);});chart.appendTo('#chart'); Property:columnWidth let chart: Chart = new Chart({ series: [{ columnWidth: 0.5 }]);});chart.appendTo('#chart'); Property:columnSpacing let chart: Chart = new Chart({ series: [{ columnSpacing: 0.5 }]);});chart.appendTo('#chart'); Property:cornerRadius.topLeft let chart: Chart = new Chart({ series: [{ topLeft: 0 }]);});chart.appendTo('#chart'); Property:cornerRadius.topRight let chart: Chart = new Chart({ series: [{ </pre>

		rectangle series	topRight: 0 });});chart.appendTo('#chart');
		bottomRight radius for rectangle series	Property:cornerRadius.bottomRight let chart: Chart = new Chart({ series: [{ bottomRight: 0 }];});chart.appendTo('#chart');
		bottomLeft radius for rectangle series	Property:cornerRadius.bottomLeft let chart: Chart = new Chart({ series: [{ bottomLeft: 0 }];});chart.appendTo('#chart');
		DashArray property	Property:dashArray let chart: Chart = new Chart({ series: [{ dashArray: '10, 5' }];});chart.appendTo('#chart');
		DataSource for series	Property:dashArray let chart: Chart = new Chart({ series: [{ dataSource: [] }];});chart.appendTo('#chart');
		Draw type for Polar series	Property:drawType let chart: Chart = new Chart({ series: [{ drawType: 'Line' }];});chart.appendTo('#chart');
		EmptyPointSettings for series	Property:emptyPointSettings.visible\$("<#chart").ejChart({ series: [{ emptyPointSettings: { visible: false } }];}); Not Applicable
		Empty Point	Property:emptyPointSettings.displayMode\$("<#chart").ejCha Property:emptyPointSettings.displayMode

		Display mode <pre>rt({ series: [{ displayMode: 'gap' }]});</pre>	<pre>let chart: Chart = new Chart({ series: [{ displayMode: 'Average' }]});chart.appe ndTo('#chart');</pre>
		Empty Point color <pre>Property:emptyPointSettings.col or\$("#chart").ejChart({ series: [{ color: 'red' }]});</pre>	Property:emptyPoint Settings.fill <pre>let chart: Chart = new Chart({ series: [{ fill: 'red' }]});chart.appe ndTo('#chart');</pre>
		Empty Point Border <pre>Property:emptyPointSettings.bo rder\$("#chart").ejChart({ series: [{ emptyPointSettings: { color: 'red', width: 2 }]});</pre>	Property:fill <pre>let chart: Chart = new Chart({ series: [{ emptyPointSettin gs: { color: 'red', width: 2 }]});chart.appe ndTo('#chart');</pre>
		Enable animation for series <pre>Property:enableAnimation\$("## chart").ejChart({ series: [{ enableAnimation: true }]});</pre>	Property:animation.e nable <pre>let chart: Chart = new Chart({ series: [animation: { enable: false }]});chart.appe ndTo('#chart');</pre>
		Animation duration for series <pre>Property:animationDuration\$("# #chart").ejChart({ series: [{ animationDuration: 1000 }]});</pre>	Property:animation.d uration <pre>let chart: Chart = new Chart({ series: [animation: { duration: 1000 }]});chart.appe ndTo('#chart');</pre>
		Animation delay for series <pre>Not Applicable</pre>	Property:animation.d uration <pre>let chart: Chart = new Chart({ series: [animation: { delay: 100 }]});chart.appe ndTo('#chart');</pre>

		<p>Drag settings for series</p> <p>Property:dragSettings\$("#chart").ejChart({ series: [{ dragSettings: { mode: 'X' } }]});</p>	Not Applicable
		<p>Errorbar settings for series</p> <p>Property:errorBarSettings\$("#chart").ejChart({ series: [{ errorBarSettings: { } }]});</p>	<p>Property:errorBarSettingslet chart: Chart = new Chart({ series: [{errorBarSettings: { } }]});</p>
		<p>Closed series</p> <p>Property:isClosed\$("#chart").ejChart({ series: [{ isClosed: true }]});</p>	<p>Property:isClosedlet chart: Chart = new Chart({ series: [{ isClosed: true }]});</p>
		<p>Stacking Property for series</p> <p>Property:isStacking\$("#chart").ejChart({ series: [{ isStacking: true }]});</p>	Not Applicable
		<p>Line cap for series</p> <p>Property:lineCap\$("#chart").ejChart({ series: [{ lineCap: 'butt' }]});</p>	Not Applicable
		<p>Line join for series</p> <p>Property:lineCap\$("#chart").ejChart({ series: [{ lineJoin: 'round' }]});</p>	Not Applicable
		<p>Opacity for series</p> <p>Property:opacity\$("#chart").ejChart({ series: [{ opacity: 0.7 }]});</p>	<p>Property:errorBarSettingslet chart: Chart = new Chart({ series: [{ opacity: 0.7 }]});</p>
		<p>Outlier settings of series</p> <p>Property:outLierSettings\$("#chart").ejChart({ series: [{ outLierSettings: { shape: 'rectangle', size: { height: 30, width: 20 } } }]});</p>	Not Applicable
		<p>Palette</p> <p>Property:palette\$("#chart").ejChart({ series: [{ palette: "ColorFieldName" }]});</p>	<p>Property:pointColorMappinglet chart: Chart = new Chart({ series: [{ pointColorMapping: "ColorFieldName" }]});</p>

		<pre>g: 'color' }}});chart.appendTo('#chart');</pre> <p>Property:pointColor Mapping let chart: Chart = new Chart({ series: [{ pointColorMapping: 'color' }]});chart.appendTo('#chart');</p> <p>Property:pointColor Mapping let chart: Chart = new Chart({ series: [{ showMean: false }]});chart.appendTo('#chart');</p> <p>Property:stackingGroup let chart: Chart = new Chart({ series: [{ stackingGroup: 'group' }]});chart.appendTo('#chart');</p> <p>Property:type let chart: Chart = new Chart({ series: [{ type: 'Line' }]});chart.appendTo('#chart');</p> <p>Property:visible let chart: Chart = new Chart({ series: [{ visible: true }]});chart.appendTo('#chart');</p> <p>Property:toggleVisibility let chart: Chart = new Chart({ legendSettings: [{</p>
	<p>Positive fill for waterfall series</p>	<pre>Property:positiveFill\$("#chart").ejChart({ series: [{ positiveFill: "red" }]});</pre>
	<p>Show average value in box and whisker series</p>	<pre>Property:showMedian\$("#chart").ejChart({ series: [{ showMedian: true }]});</pre>
	<p>To group the series of stacking collection.</p>	<pre>Property:stackingGroup\$("#chart").ejChart({ series: [{ stackingGroup: 'group' }]});</pre>
	<p>Specifies the type of the series to render in chart.</p>	<pre>Property:type\$("#chart").ejChart({ series: [{ type: 'Line' }]});</pre>
	<p>Defines the visibility of the series.</p>	<pre>Property:visibility\$("#chart").ejChart({ series: [{ visibility: true }]});</pre>
	<p>Enables or disables the visibility</p>	<pre>Property:visibleOnLegend\$("#chart").ejChart({ series: [{ visibleOnLegend : true }]});</pre>

		<p>of legend item.</p> <p>Specifies the different types of spline curve.</p> <p>Specifies the name of the x-axis that has to be associated with this series. Add an axis instance with this name to axes collection.</p> <p>Name of the property in the datasource that contains x value for the series.</p> <p>Specifies the name of the y-axis that</p>	<pre>toggleVisibility : true });});chart.appendTo('#chart');</pre> <p>Property:splineType1</p> <pre>let chart: Chart = new Chart({ legendSettings: [{ splineType: 'Natural' }]);});chart.appendTo('#chart');</pre> <p>Property:xAxisName</p> <pre>let chart: Chart = new Chart({ series: [{ xAxisName: 'secondaryXAxis' }]);});chart.appendTo('#chart');</pre> <p>Property:xName</p> <pre>let chart: Chart = new Chart({ series: [{ xName: 'x' }]);});chart.appendTo('#chart');</pre> <p>Property:yAxisName</p> <pre>let chart: Chart = new Chart({ series: [{ yAxisName: 'secondaryYAxis' }</pre>
--	--	---	--

		<p>has to be associated with this series. Add an axis instance with this name to axes collection.</p> <p>Name of the property in the datasource that contains y value for the series.</p> <p>Name of the property in the datasource that contains high value for the series.</p> <p>Name of the property in the datasource that contains low value for</p>	<pre>]});chart.appendTo('#chart'); Property:yNamelet chart: Chart = new Chart({ series: [{ yName: 'y' }]);chart.appendTo('#chart'); Property:highlet chart: Chart = new Chart({ series: [{ high: 'y' }]);chart.appendTo('#chart'); Property:lowlet chart: Chart = new Chart({ series: [{ low: 'y' }]);chart.appendTo('#chart'); </pre>
--	--	--	---

		<p>the series.</p> <p>Name of the property in the data source that contains close value for the series.</p> <p>Name of the property in the data source that contains open value for the series.</p> <p>Option to add trendline series to chart.</p> <p>Options for customizing the appearance of the series or data point while highlighting.</p>	<p>Property:close <pre>let chart: Chart = new Chart({ series: [{ close: 'y' }] }); chart.appendTo('#chart');</pre></p> <p>Property:open <pre>let chart: Chart = new Chart({ series: [{ open: 'y' }] }); chart.appendTo('#chart');</pre></p> <p>Property:trendLines <pre>let chart: Chart = new Chart({ series: [{ trendLines : [{}] }] }); chart.appendTo('#chart');</pre></p> <p>Property:highlightSettings <pre>let chart: Chart = new Chart({ series: [{ highlightSettings : {} }] }); chart.appendTo('#chart');</pre></p> <p>Not applicable.</p>
--	--	---	--

		Options for customizing the appearance of the series/data point on selection .	
		Property:selectionSettings <pre>\$("#chart").ejChart({ series: [{ selectionSettings : {} }];});</pre>	Not applicable.
	visibility of marker	Property:visible <pre>\$("#chart").ejChart({ series: [{ marker: { visible: true } }];});</pre>	Property:visible <pre>let chart: Chart = new Chart({ series: [{ marker: { visible: false } }];});chart.appendTo('#chart');</pre>
	Fill for marker	Property:fill <pre>\$("#chart").ejChart({ series: [{ marker: { fill : 'red' } }];});</pre>	Property:visible <pre>let chart: Chart = new Chart({ series: [{ marker: { fill : 'red' } }];});chart.appendTo('#chart');</pre>
	Opacity for marker	Property:opacity <pre>\$("#chart").ejChart({ series: [{ marker: { opacity : 0.5 } }];});</pre>	Property:opacity <pre>let chart: Chart = new Chart({ series: [{ marker: { opacity : 0.5 } }];});chart.appendTo('#chart');</pre>
	Shape of marker	Property:shape <pre>\$("#chart").ejChart({ series: [{ marker: { shape : 'Circle' } }];});</pre>	Property:shape <pre>let chart: Chart = new Chart({ series: [{ marker: { shape : 'Triangle' } }];});chart.appendTo('#chart');</pre>

		<p>Image Url of marker</p> <pre>Property:imageUrl\$("#chart").ejChart({ series: [{ marker: { imageUrl : '' } }]});</pre>	<pre>Property:imageUrllet chart: Chart = new Chart({ series: [{ marker: { imageUrl : '' } }]});chart.appendTo('#chart');</pre>
		<p>Border of marker</p> <pre>Property:border\$("#chart").ejChart({ series: [{ marker: { border : { width: 2, color: 'red' } } }]});</pre>	<pre>Property:shapelet chart: Chart = new Chart({ series: [{ marker: { border : { width: 2, color: 'red' } } }]});chart.appendTo('#chart');</pre>
		<p>Height of marker</p> <pre>Property:size.height\$("#chart").ejChart({ series: [{ marker: { size: { height: 30 } } }]});</pre>	<pre>Property:heightlet chart: Chart = new Chart({ series: [{ marker: { height: 25 } }]});chart.appendTo('#chart');</pre>
		<p>Width of marker</p> <pre>Property:size.width\$("#chart").ejChart({ series: [{ marker: { size: { width: 30 } } }]});</pre>	<pre>Property:widthlet chart: Chart = new Chart({ series: [{ marker: { width: 25 } }]});chart.appendTo('#chart');</pre>
		<p>Width of marker</p> <pre>Property:size.width\$("#chart").ejChart({ series: [{ marker: { size: { width: 30 } } }]});</pre>	<pre>Property:widthlet chart: Chart = new Chart({ series: [{ marker: { width: 25 } }]});chart.appendTo('#chart');</pre>
		<p>Data Labels of marker</p> <pre>Property:marker.dataLabel\$("#chart").ejChart({ series: [{ marker: { dataLabel: { } } }]});</pre>	<pre>Property:marker.dataLabellet chart: Chart = new Chart({ series: [{ marker: { dataLabel: { } } }]});</pre>

		<pre> } }};});chart.appendTo('#chart'); Property:dataLabel. visiblelet chart: Chart = new Chart({ series: [[marker: { dataLabel: { visible: true } }]}); }};});chart.appendTo('#chart'); Property:dataLabel. namelet chart: Chart = new Chart({ series: [[marker: { dataLabel: { name: '' } } }};});chart.appendTo('#chart'); Property:dataLabel. filllet chart: Chart = new Chart({ series: [[marker: { dataLabel: { fill: 'pink' } } }};});chart.appendTo('#chart'); Property:dataLabel. filllet chart: Chart = new Chart({ series: [[marker: { dataLabel: { opacity: 0.4 } }};});chart.appendTo('#chart'); Property:dataLabel. positionlet chart: Chart = new Chart({ series: [[marker: { dataLabel: { position: 'Top' </pre>
Visibility of dataLabel	<pre> Property:dataLabel.visible\$("#chart").ejChart({ series: [[marker: {dataLabel: { visible: true } }]}); </pre>	
Text mapping name of dataLabel	<pre> Property:dataLabel.textMappingName\$("#chart").ejChart({ series: [{ marker: {dataLabel: { textMappingName: '' } } }]}); </pre>	
Fill color of data label	<pre> Property:dataLabel.fill\$("#chart").ejChart({ series: [{ marker: {dataLabel: { fill: 'pink' } } }]}); </pre>	
Opacity of data label	<pre> Property:dataLabel.opacity\$("#chart").ejChart({ series: [[marker: {dataLabel: { opacity: 0.6 } }]}); </pre>	
Text position of data label	<pre> Property:dataLabel.textPosition\$("#chart").ejChart({ series: [{ marker: {dataLabel: { textPosition: 'middle' } } }]}); </pre>	

		<pre> } } }};});chart.append endTo('#chart); </pre>
Alignment of data label	Property:dataLabel.verticalAlignment <pre> ent\$("#chart").ejChart({ series: [{ marker: {dataLabel: { verticalAlignment: 'near' } } }];}); </pre>	Property:dataLabel.alignment <pre> let chart: Chart = new Chart({ series: [{ marker: { dataLabel: { alignment: 'Near' } } }];});chart.append endTo('#chart); </pre>
Border of data label	Property:dataLabel.border <pre> \$("#chart").ejChart({ series: [{ marker: {dataLabel: { border: { color: 'blue', width: 2, opacity: 0.4 } } } }];}); </pre>	Property:dataLabel.alignment <pre> let chart: Chart = new Chart({ series: [{ marker: { dataLabel: { border: { color: 'blue', width: 2 } } } }];});chart.append endTo('#chart); </pre>
Offset for data label	Property:dataLabel.offset <pre> \$("#chart").ejChart({ series: [{ marker: {dataLabel: { offset: { x: 5, y: 6 } } } }];}); </pre>	Not Applicable
Margin of data label	Property:dataLabel.border <pre> \$("#chart").ejChart({ series: [{ marker: {dataLabel: { margin: { top: 10, bottom: 10, left: 10, right: 10} } } }];}); </pre>	Property:dataLabel.margin <pre> let chart: Chart = new Chart({ series: [{ marker: { dataLabel: { margin: { top: 10, bottom: 10, left: 10, right: 10 } } } }];});chart.append endTo('#chart); </pre>
Font of data label	Property:dataLabel.border <pre> \$("#chart").ejChart({ series: [{ marker: {dataLabel: { font: { fontFamily: 'SegoeUI', fontStyle: 'italic', fontWeight: </pre>	Property:dataLabel.margin <pre> let chart: Chart = new Chart({ series: [{ marker: </pre>

		<pre>'600', opacity: 0.5, size: 12, color: 'red' }} } }};));</pre>	<pre>{dataLabel: { font: { fontFamily: 'SegoeUI', fontStyle: 'italic', fontWeight: '600', opacity: 0.5, size: 12, color: 'red' }} } }];});chart.append endTo('#chart);</pre>
HTML templat e in dataLab el	Property:dataLabel.template\$(" chart").ejChart({ series: [{ marker: {dataLabel: { template: ' Chart ' } } }];});	new Chart({ series: [{ marker: {dataLabel: { template: ' Chart ' } } }];});chart.append endTo('#chart);	Property:dataLabel. templatelet chart: Chart = new Chart({ series: [{ marker: {dataLabel: { template: ' Chart ' } } }];});chart.append endTo('#chart);
Rounde d corner radius X	Not Applicable	rxlet chart: Chart = new Chart({ series: [{ marker: {dataLabel: { rx: 10 } } }];});chart.append endTo('#chart);	Property:dataLabel. rxlet chart: Chart = new Chart({ series: [{ marker: {dataLabel: { rx: 10 } } }];});chart.append endTo('#chart);
Rounde d corner radius Y	Not Applicable	rylet chart: Chart = new Chart({ series: [{ marker: {dataLabel: { ry: 10 } } }];});chart.append endTo('#chart);	Property:dataLabel. rylet chart: Chart = new Chart({ series: [{ marker: {dataLabel: { ry: 10 } } }];});chart.append endTo('#chart);
Maximu m Label width for data label	Property:dataLabel.maximumLabe lWidth\$(" chart").ejChart({ series: [{ marker: {dataLabel: { maximumLabelWidth: 20}} } }];});	Not Applicable	

		<p>Enable wrapping of text for data label</p> <p>Property: dataLabel.enableWrap\$</p> <pre>("#chart").ejChart({ series: [{ marker: {dataLabel: { enableWrap: true }} } }];});</pre> <p>Not Applicable</p> <p>To show contrast color for data label</p> <p>Property: dataLabel.showContrastColor\$</p> <pre>("#chart").ejChart({ series: [{ marker: {dataLabel: { showContrastColor: true }} } }];});</pre> <p>Not Applicable</p> <p>To show edge label for data label</p> <p>Property: dataLabel.showEdgeLabels\$</p> <pre>("#chart").ejChart({ series: [{ marker: {dataLabel: { showEdgeLabels: true }} } }];});</pre> <p>Not Applicable</p> <p>## TrendLines</p> <p>Behaviour</p> <p>API in Essential JS 1</p> <p>API in Essential JS 2</p> <p>Trend lines settings</p> <p>Property: series.trendLines\$</p> <pre>let chart: Chart = new Chart({ series: [{ trendLines: [] }]});chart.appendTo('#chart');</pre> <p>Visibility of trend line</p> <p>Property: trendLines.visibility\$</p> <pre>("#chart").ejChart({ series: [{ trendLines: [visibility: true] }]});</pre> <p>Not applicable</p> <p>Type of trend line</p> <p>Property: trendLines.type\$</p> <pre>("#chart").ejChart({ series: [{ trendLines: [type: 'linear'] }]});</pre> <p>Property: trendLines.type\$</p> <pre>let chart: Chart = new Chart({ series: [{ trendLines: [type: 'Polynomial'] }]});chart.appendTo('#chart');</pre> <p>Name of trend line</p> <p>Property: trendLines.name\$</p> <pre>("#chart").ejChart({ series: [{ trendLines: [name: 'trendLine'] }]});</pre> <p>Property: trendLines.name\$</p> <pre>let chart: Chart = new Chart({ series: [{ trendLines: [name: 'trendLine'] }]});cha</pre>
--	--	--

		<pre>rt.appendTo('#chart'));</pre>
Period of trendLine	<pre>Property:trendLines.period\$ (" #chart").ejChart({ series: [{ trendLines: [period: 45]}]})</pre>	<pre>Property:trendLines.period let chart: Chart = new Chart({ series: [{ trendLines: [period: 45]})});chart.append To('#chart');</pre>
Polynomial order for Polynomial trendLines	<pre>Property:trendLines.polynomialOrder\$ ("#chart").ejChart({ series: [{ trendLines: [polynomialOrder: 3]}]})</pre>	<pre>Property:trendLines.polynomialOrder let chart: Chart = new Chart({ series: [{ trendLines: [polynomialOrder: 3]})});chart.appendT o('#chart');</pre>
Backward forecast trendLines	<pre>Property:trendLines.backwardforecast\$ ("#chart").ejChart({ series: [{ trendLines: [backwardforecast: 3]}]})</pre>	<pre>Property:trendLines.backwardforecast let chart: Chart = new Chart({ series: [{ trendLines: [backwardforecast: 3]})});chart.appendT o('#chart');</pre>
Forward forecast trendLines	<pre>Property:trendLines.forwardForecast\$ ("#chart").ejChart({ series: [{ trendLines: [forwardForecast: 3]}]})</pre>	<pre>Property:trendLines.forwardForecast let chart: Chart = new Chart({ series: [{ trendLines: [forwardForecast: 3]})});chart.appendT o('#chart');</pre>
Fill for trendLines	<pre>Property:trendLines.fill\$ ("#chart").ejChart({ series: [{ trendLines: [fill: 'EEFFCC']}]})</pre>	<pre>Property:trendLines.fill let chart: Chart = new Chart({ series: [{ trendLines: [fill: 'EEFFCC']})});chart. appendTo('#chart');</pre>
Width for	<pre>Property:trendLines.width\$ (" #chart").ejChart({</pre>	<pre>Property:trendLines.width let chart: Chart =</pre>

		<pre> trendLines series: [{ trendLines: [width: 2]}]}}); </pre>	<pre> new Chart({ series: [{ trendLines: [width: 2]}]});chart.appendT o('#chart'); </pre>
		<pre> Intercept Property:trendLines.intercept\$ let chart: Chart = new Chart({ series: [{ trendLines: [intercept: 2]}]}}); </pre>	<pre> Property:trendLines.intercept\$ let chart: Chart = new Chart({ series: [{ trendLines: [intercept: 2]}]});chart.appendT o('#chart'); </pre>
		<pre> LegendShape Not Applicable </pre>	<pre> Property:trendLines.legendShape let chart: Chart = new Chart({ series: [{ trendLines: [legendShape: 'Rectangle']}]});cha rt.appendTo('#chart'); </pre>
		<pre> AnimationSettings Not Applicable </pre>	<pre> Property:trendLines.animation let chart: Chart = new Chart({ series: [{ trendLines: [animation: { enable: true }}]}]);chart.appendT o('#chart'); </pre>
		<pre> MarkerSettings Not Applicable </pre>	<pre> Property:trendLines.marker let chart: Chart = new Chart({ series: [{ trendLines: [{marker: { visible: true }}]}]});chart.append To('#chart'); </pre>
		<pre> Tooltip Property:trendLines.tooltip\$ (" #chart").ejChart({ series: [{ trendLines: [{ tooltip: { } }]}]}); </pre>	<pre> Property:trendLines.enableTooltip let chart: Chart = new Chart({ series: [{ trendLines: [{enableTooltip: true }}]}]);chart.appendT o('#chart'); </pre>

		<p>Dash</p> <p>Array for trendLines</p> <p>Property: trendLines.dashArray</p> <pre>\$("#chart").ejChart({ series: [{ trendLines: [{ dashArray: '10, 5' }] }]});</pre> <p>Not Applicable.</p> <p>Visible on legend</p> <p>Property: trendLines.visibleOnLegend</p> <pre>\$("#chart").ejChart({ series: [{ trendLines: [{ visibleOnLegend: true }] }]});</pre> <p>Not Applicable.</p> <p>## Striplines</p> <p>Behaviour</p> <p>API in Essential JS 1</p> <p>API in Essential JS 2</p> <p>Default behaviour for striplines</p> <p>Property: primaryXAxis.striplines</p> <pre>let chart: Chart = new Chart({ primaryXAxis: { striplines: [{ visible: true }] } });chart.appendTo('#chart');</pre> <p>border for stripline</p> <p>Property: striplines.borderColor</p> <pre>let chart: Chart = new Chart({ primaryXAxis: { striplines: [{ border: { color: 'red', width: 2 } }] });chart.appendTo('#chart');</pre> <p>Background color for stripline</p> <p>Property: striplines.borderColor</p> <pre>let chart: Chart = new Chart({ primaryXAxis: { striplines: [{ color: 'red' }] });chart.appendTo('#chart');</pre> <p>Start value for</p> <p>Property: striplines.start</p> <pre>let chart: Chart = new Chart({ primaryXAxis: {</pre>
--	--	---

		<pre>stripLines: [{ start: stripLines: [{ 10 }]]}}]; start: 5}}]}}];chart.appendTo('#chart');</pre>
End value for stripLine	<pre>Property:stripLines.end\$("#chart").ejChart({ primaryXAxis: { stripLines: [{ end: 10 }]]}}];</pre>	<pre>Property:stripLines.endlet chart: Chart = new Chart({ primaryXAxis: { stripLines: [{ end: 5}}]}}];chart.appendTo('#chart');</pre>
Start from Axis for stripLine	<pre>Property:stripLines.startFromAxis\$("#chart").ejChart({ primaryXAxis: { stripLines: [{ startFromAxis: true }}]}}];</pre>	<pre>Property:stripLines.startFromAxislet chart: Chart = new Chart({ primaryXAxis: { stripLines: [{ startFromAxis: true }}]}}];chart.appendTo('#chart');</pre>
Text in stripLine	<pre>Property:stripLines.text\$("#chart").ejChart({ primaryXAxis: { stripLines: [{ text: 'StripLine; }]]}}];</pre>	<pre>Property:stripLines.textlet chart: Chart = new Chart({ primaryXAxis: { stripLines: [{ text: 'StripLine; }]]}}];chart.appendTo('#chart');</pre>
Text alignment in stripLine	<pre>Property:stripLines.textAlignment\$("#chart").ejChart({ primaryXAxis: { stripLines: [{ textAlignment: 'Far; }}]}}];</pre>	<pre>Property:stripLines.horizontalAlignmentlet chart: Chart = new Chart({ primaryXAxis: { stripLines: [{ horizontalAlignment: 'Far; }]]}}];chart.appendTo('#chart');</pre>
Vertical Text alignment in stripLine	Not Applicable	<pre>Property:stripLines.verticalAlignmentlet chart: Chart = new Chart({ primaryXAxis: { stripLines: [{ verticalAlignment: 'Far; }]]}}];chart.appendTo('#chart');</pre>
Size of stripLine	<pre>Property:stripLines.width\$("#chart").ejChart({ primaryXAxis: { stripLines: [{ width: 10; }]]}}];</pre>	<pre>Property:stripLines.sizelet chart: Chart = new Chart({ primaryXAxis: { stripLines: [{</pre>

		<pre>size: 10 }}}});chart.appendTo ('#chart');</pre>
ZIndex of stripLine	<pre>Property:stripLines.zIndex\$("#chart").ejChart({ primaryXAxis: { stripLines: [{ zIndex: 'Behind' }}}});</pre>	<pre>Property:stripLines.sizele t chart: Chart = new Chart({ primaryXAxis: { stripLines: [{ zIndex: 'Behind' }}}});chart.appendTo ('#chart');</pre>
Font style of stripLine	<pre>Property:stripLines.fontStyle\$("#chart").ejChart({ primaryXAxis: { stripLines: [{ fontStyle: {} }]]}});</pre>	<pre>Property:stripLines.textStyl elet chart: Chart = new Chart({ primaryXAxis: { stripLines: [{ textStyle: {} }}}});chart.appendTo ('#chart');</pre>
## Multilevel Labels		
Behavior	API in Essential JS 1	API in Essential JS 2
Default behavior for multilevels	<pre>Property:primaryXAxis.multilev elLabels\$("#chart").ejCha rt({ primaryXAxis: { multilevelLabels: [{ visible: true }]]}});</pre>	<pre>Property:primaryXAxis.m ultilevelLabelslet chart: Chart = new Chart({ primaryXAxis: { multilevelLabels: [{ }}}});chart.appendT o('#chart');</pre>
Default behavior for multilevels	<pre>Property:primaryXAxis.multilev elLabels\$("#chart").ejCha rt({ primaryXAxis: { multilevelLabels: [{ visible: true }]]}});</pre>	<pre>Property:primaryXAxis.m ultilevelLabelslet chart: Chart = new Chart({ primaryXAxis: { multilevelLabels: [{ }}}});chart.appendT o('#chart');</pre>
Text alignment for multil	<pre>Property:multiLevellLabels.text Alignment\$("#chart").ejC hart({ primaryXAxis: { multilevelLabels: [{ textAlignment: 'Near' }}}});</pre>	<pre>Property:multilevelLabels .alignmentlet chart: Chart = new Chart({ primaryXAxis: { multilevelLabels: [{alignment: 'Near'</pre>

		<pre> evelLa bels }]]]]);chart.appendT o('#chart'); Property:multiLevelLabel s.overflowlet chart: OverFlow\$("#chart").ejCh art({ primaryXAxis: { multilevelLabels: [{ textOverFlow: 'Trim' }}]]]]); }]]]]);chart.appendT o('#chart'); Property:multiLevelLabel s.borderlet chart: der\$("#chart").ejChart({ primaryXAxis: { multilevelLabels: [{ border: { width: 2, color: 'red', type: 'brace' } }]]]]); }]]]]);chart.appendT o('#chart'); Property:multiLevelLabel s.categories.startlet chart: Chart = new Chart({ primaryXAxis: { multilevelLabels: [{ categories: [{ start: 45}]] } }}]]];chart.appendT o('#chart'); Property:multiLevelLabel s.categories.endlet chart: Chart = new Chart({ primaryXAxis: { multilevelLabels: [{ categories: [{ end: 45}]] } }}]]];chart.appendT o('#chart'); Property:multiLevelLabel s.categories.textlet chart: Chart = new Chart({ primaryXAxis: { multilevelLabels: [{ categories: [{ text: 'text' }]] } } </pre>
	<p>Text overfl ow for multil evelLa bels</p>	<pre> Property:multiLevelLabels.text OverFlow\$("#chart").ejCh art({ primaryXAxis: { multilevelLabels: [{ textOverFlow: 'Trim' }}]]]]); }]]]]);chart.appendT o('#chart'); Property:multiLevelLabel s.overflowlet chart: OverFlow\$("#chart").ejCh art({ primaryXAxis: { multilevelLabels: [{ textOverFlow: 'Trim' }}]]]]); }]]]]);chart.appendT o('#chart'); Property:multiLevelLabel s.borderlet chart: der\$("#chart").ejChart({ primaryXAxis: { multilevelLabels: [{ border: { width: 2, color: 'red', type: 'brace' } }]]]]); }]]]]);chart.appendT o('#chart'); Property:multiLevelLabel s.categories.startlet chart: Chart = new Chart({ primaryXAxis: { multilevelLabels: [{ categories: [{ start: 45}]] } }}]]];chart.appendT o('#chart'); Property:multiLevelLabel s.categories.endlet chart: Chart = new Chart({ primaryXAxis: { multilevelLabels: [{ categories: [{ end: 45}]] } }}]]];chart.appendT o('#chart'); Property:multiLevelLabel s.categories.textlet chart: Chart = new Chart({ primaryXAxis: { multilevelLabels: [{ categories: [{ text: 'text' }]] } } </pre>
	<p>Borde r for multil evelLa bels</p>	<pre> Property:multiLevelLabels.bor der\$("#chart").ejChart({ primaryXAxis: { multilevelLabels: [{ border: { width: 2, color: 'red', type: 'brace' } }]]]]); }]]]]);chart.appendT o('#chart'); Property:multiLevelLabel s.borderlet chart: der\$("#chart").ejChart({ primaryXAxis: { multilevelLabels: [{ border: { width: 2, color: 'red', type: 'brace' } }]]]]); }]]]]);chart.appendT o('#chart'); Property:multiLevelLabel s.categories.startlet chart: Chart = new Chart({ primaryXAxis: { multilevelLabels: [{ categories: [{ start: 45}]] } }}]]];chart.appendT o('#chart'); Property:multiLevelLabel s.categories.endlet chart: Chart = new Chart({ primaryXAxis: { multilevelLabels: [{ categories: [{ end: 45}]] } }}]]];chart.appendT o('#chart'); Property:multiLevelLabel s.categories.textlet chart: Chart = new Chart({ primaryXAxis: { multilevelLabels: [{ categories: [{ text: 'text' }]] } } </pre>
	<p>Start value for label</p>	<pre> Property:multiLevelLabels.start \$("#chart").ejChart({ primaryXAxis: { multilevelLabels: [{ start: 45 } }]]]]); }]]]]);chart.appendT o('#chart'); Property:multiLevelLabel s.categories.startlet chart: Chart = new Chart({ primaryXAxis: { multilevelLabels: [{ categories: [{ start: 45}]] } }}]]];chart.appendT o('#chart'); Property:multiLevelLabel s.categories.endlet chart: Chart = new Chart({ primaryXAxis: { multilevelLabels: [{ categories: [{ end: 45}]] } }}]]];chart.appendT o('#chart'); Property:multiLevelLabel s.categories.textlet chart: Chart = new Chart({ primaryXAxis: { multilevelLabels: [{ categories: [{ text: 'text' }]] } } </pre>
	<p>End value for label</p>	<pre> Property:multiLevelLabels.start \$("#chart").ejChart({ primaryXAxis: { multilevelLabels: [{ end: 45 } }]]]]); }]]]]);chart.appendT o('#chart'); Property:multiLevelLabel s.categories.startlet chart: Chart = new Chart({ primaryXAxis: { multilevelLabels: [{ categories: [{ start: 45}]] } }}]]];chart.appendT o('#chart'); Property:multiLevelLabel s.categories.endlet chart: Chart = new Chart({ primaryXAxis: { multilevelLabels: [{ categories: [{ end: 45}]] } }}]]];chart.appendT o('#chart'); Property:multiLevelLabel s.categories.textlet chart: Chart = new Chart({ primaryXAxis: { multilevelLabels: [{ categories: [{ text: 'text' }]] } } </pre>
	<p>Text for label</p>	<pre> Property:multiLevelLabels.text \$("#chart").ejChart({ primaryXAxis: { multilevelLabels: [{ text: 'Start' } }]]]]); }]]]]);chart.appendT o('#chart'); Property:multiLevelLabel s.categories.textlet chart: Chart = new Chart({ primaryXAxis: { multilevelLabels: [{ categories: [{ text: 'text' }]] } } </pre>

		<pre> }}}});chart.appendTo('#chart'); Property:multiLevelLabels.categories.maximumTextWidth let chart: Chart = new Chart({ primaryXAxis: { multilevelLabels: [{ categories: [{ maximumTextWidth: 20 }] } }] } });chart.appendTo('#chart'); </pre>
maximum text width for label	<pre> Property:multiLevelLabels.maximumTextWidth\$("#chart").ejChart({ primaryXAxis: { multilevelLabels: [{ maximumTextWidth: 10 } }] } }); </pre>	<pre> Property:multiLevelLabels.categories.categories[0].maximumTextWidth\$("#chart").ejChart({ primaryXAxis: { multilevelLabels: [{ maximumTextWidth: 20 }] } }] } });chart.appendTo('#chart'); </pre>
level of labels	<pre> Property:multiLevelLabels.level l\$("#chart").ejChart({ primaryXAxis: { multilevelLabels: [{ level: 2 }] } } }); </pre>	<pre> Property:multiLevelLabels.level Not applicable. Categories are used </pre>
## Methods		
Behaviour	API in Essential JS 1	API in Essential JS 2
animation for series	<pre> Property:chart.animate\$("#chart").ejChart({ animate: () { } }); </pre>	<pre> Property:chart.animate\$("#chart").ejChart({ animate: () { } }); Not applicable </pre>
Redraw for chart	<pre> Property:chart.redraw\$("#chart").ejChart({ redraw: () { } }); </pre>	<pre> Property:chart.refresh()let chart: Chart = new Chart({});chart.appendTo('#chart');chart.width = '400';chart.refresh(); </pre>
Export	<pre> Property:chart.export()\$("#chart").ejChart({ export: () { } }); </pre>	<pre> Property:chart.export()let chart: Chart = new Chart({});chart.export('J PEG', 'chart');chart.appendTo('#chart'); </pre>
Print	<pre> Property:chart.print()\$("#chart").ejChart({ print: () { } }); </pre>	<pre> Property:chart.print()let chart: Chart = new Chart({});chart.print('chart');chart.appendTo('#chart'); </pre>
AddSeries	Not Applicable	<pre> Property:chart.addSeries()let chart: Chart = new Chart({});chart.appendTo(</pre>

		<pre>'#chart');chart.addSeries ();</pre>
Removes Series	Not Applicable	<pre>Property:chart.removeSeries()let t chart: Chart = new Chart({});chart.appendTo('#chart');chart.removeSer ies();</pre>
## Events		
Behavior	API in Essential JS 1	API in Essential JS 2
Fires on annotation click	<pre>Property:annotationClick\$("#cha rt").ejChart({ annotationClick: () { }});</pre>	Not applicable
Fires after animation	<pre>Property:animationComplete\$("# chart").ejChart({ animationComplete: () { }});</pre>	<pre>Property:animationC omplete()let chart: Chart = new Chart({ animationComple e: () => { }});chart.append To('#chart');</pre>
Fires on axis label click	<pre>Property:axisLabelClick\$("#chart ").ejChart({ axisLabelClick: () { }});</pre>	Not applicable
Fires before axis label render	<pre>Property:axisLabelRendering\$("#c hart").ejChart({ axisLabelRendering: () { }});</pre>	<pre>Property:axisLabelRe nder()let chart: Chart = new Chart({ axisLabelRender: () => { }});chart.append To('#chart');</pre>
Fires on axis label mouse move	<pre>Property:axisLabelMouseMove\$ (" #chart").ejChart({ axisLabelMouseMove: () { }});</pre>	Not applicable
Fires on	<pre>Property:axisLabelInitialize\$ ("#ch art").ejChart({</pre>	Not applicable

		<pre> axis axisLabelInitialize: () { label }}); initiali ze </pre>
		<pre> Fires before Property:axesRangeCalculate\$("# axis chart").ejChart({ range axesRangeCalculate: () { calcula }}); tion </pre>
		<pre> Fires on Property:axisTitleRendering\$("#c axis hart").ejChart({ title axisTitleRendering: () { render }}); ing </pre>
		<pre> Fires on Property:afterResize\$("#chart") after .ejChart({ afterResize: () chart { }}); resize </pre>
		<pre> Fires on Property:beforeResize\$("#chart before ") .ejChart({ beforeResize: chart () { }}); resize </pre>
		<pre> Fires on Property:chartClick\$("#chart"). chart ejChart({ chartClick: () { click }}); </pre>
		<pre> Fires on Property:chartMouseMove\$("#ch chart art").ejChart({ mouse chartMouseMove: () { }}); move </pre>

		<p>Fires on chart mouse leave</p> <p>Property:chartMouseLeave\$("#chart").ejChart({chartMouseLeave: () { }});</p>	<p>Property:chartMouseLeave</p> <pre>let chart: Chart = new Chart({ chartMouseLeave: () => { }});chart.append To('#chart');</pre>
		<p>Fires on chart double click</p> <p>Property:chartDoubleClick\$("#chart").ejChart({chartDoubleClick: () { }});</p>	Not applicable
		<p>Fires on chart mouse up</p> <p>Not Applicable</p>	<p>Property:chartmouseUp</p> <pre>let chart: Chart = new Chart({ chartmouseUp: () => { }});chart.append To('#chart');</pre>
		<p>Fires on chart mouse down</p> <p>Not Applicable</p>	<p>Property:chartmouseDown</p> <pre>let chart: Chart = new Chart({ chartmouseDown: () => { }});chart.append To('#chart');</pre>
		<p>Fires during the calculation of chart area bounds. You can use this event to customize the</p> <p>Property:chartAreaBoundsCalculate\$("#chart").ejChart({chartAreaBoundsCalculate: () { }});</p>	Not applicable

		<p>bound s of chart area</p> <p>Fires when the draggi ng is starte d</p> <p>Fires while draggi ng</p> <p>Fires when the draggi ng is compl eted</p> <p>Fires when chart is destro yed compl etely.</p> <p>Fires after chart is create d.</p> <p>Fires before render ing the</p>	<pre> Property:dragStart\$("#chart"). ejChart({ dragStart: () { Not applicable }}); Property:dragging\$("#chart"). ejChart({ dragging: () { Not applicable }}); Property:dragCompletelet chart: Chart = new Chart({ dragComplete: () => { }});chart.append To('#chart'); Property:destroy\$("#chart").e jChart({ destroy: () { Not applicable }}); Property:loadedlet chart: Chart = new Chart({ loaded: () => { }});chart.append To('#chart'); Property:textRender let chart: Chart = new Chart({ textRender: () => { }});chart.append To('#chart'); </pre>
--	--	---	--

		<p>data labels.</p> <p>Fires when error bar is rendering.</p> <p>Property: <code>errorBarRendering\$ ("#chart").ejChart({ errorBarRendering: () { }}</code> Not applicable</p> <p>Fires during the calculation of legend bounds.</p> <p>Property: <code>legendBoundsCalculate\$ ("#chart").ejChart({ legendBoundsCalculate: () { }}</code> Not applicable</p> <p>Fires on clicking the legend item.</p> <p>Property: <code>legendItemClick\$ ("#chart").ejChart({ legendItemClick: () { }}</code> Not applicable</p> <p>Fires when moving mouse over legend item.</p> <p>Property: <code>legendItemMouseMove\$ ("#chart").ejChart({ legendItemMouseMove: () { }}</code> Not applicable</p> <p>Fires before rendering the legend item.</p> <p>Property: <code>legendItemRendering\$ ("#chart").ejChart({ legendItemRendering: () { }}</code> Property: legendRender <pre>erlet chart: Chart = new Chart({ legendRender: () => { }});chart.append To('#chart');</pre></p> <p>Fires before loading the chart.</p> <p>Property: <code>load\$ ("#chart").ejChart({ load: () { }}</code> Property: load <pre>let chart: Chart = new Chart({ load: () => { }});chart.append To('#chart');</pre></p>
--	--	--

		<p>Fires, when multi level labels are rendering.</p> <p>Property:multiLevelLabelRendering <pre>\$("#chart").ejChart({ multiLevelLabelRendering: () { }});</pre> </p> <p>Fires on clicking a point in chart.</p> <p>Property:pointRegionClick <pre>\$("#chart").ejChart({ pointRegionClick: () { }});</pre> </p> <p>Fires when mouse is moved over a point.</p> <p>Property:pointRegionMouseMove <pre>\$("#chart").ejChart({ pointRegionMouseMove: () { }});</pre> </p> <p>Fires before rendering chart.</p> <p>Property:preRender <pre>\$("#chart").ejChart({ preRender: () { }});</pre> </p> <p>Fires when point render .</p> <p>Property:pointRender <pre>let chart: Chart = new Chart({ pointRender : () => { }});chart.append To('#chart');</pre> </p> <p>Fires after selected the data in chart.</p> <p>Property:rangeSelected <pre>\$("#chart").ejChart({ rangeSelected: () { }});</pre> </p> <p>Fires after selecti</p> <p>Property:seriesRegionClick <pre>\$("#chart").ejChart({</pre> </p>	<p>Property:axisMultiLabelRender <pre>let chart: Chart = new Chart({ axisMultiLabelRender : () => { }});chart.append To('#chart');</pre> </p> <p>Property:pointClick <pre>let chart: Chart = new Chart({ pointClick : () => { }});chart.append To('#chart');</pre> </p> <p>Property:pointMove <pre>let chart: Chart = new Chart({ pointMove : () => { }});chart.append To('#chart');</pre> </p> <p>Not applicable</p> <p>Property:pointRender <pre>let chart: Chart = new Chart({ pointRender : () => { }});chart.append To('#chart');</pre> </p> <p>Not applicable</p> <p>Not applicable</p>
--	--	---	---

		<pre> ng a seriesRegionClick: () { series. }}}; </pre>	
		<pre> Fires before Property:seriesRendering\$ ("#chart").ejChart({ render rt").ejChart({ ing a seriesRendering: () { }}}; series. </pre>	<p>Property:seriesRender</p> <pre> er let chart: Chart = new Chart({ seriesRender : () => { }});chart.append To('#chart'); </pre>
		<pre> Fires before render ing Property:symbolRendering\$ ("#chart").ejChart({ the art").ejChart({ marke symbolRendering: () { }}}; r symbo ls. </pre>	<p>Not applicable</p>
		<pre> Fires before render Property:trendlineRendering\$ ("#chart").ejChart({ ing chart").ejChart({ the trendlineRendering: () { trendli }}}; ne </pre>	<p>Not applicable</p>
		<pre> Fires before render Property:titleRendering\$ ("#chart").ejChart({ ing t").ejChart({ the titleRendering: () { }}}; Chart title. </pre>	<p>Not applicable</p>
		<pre> Fires before render Property:subTitleRendering\$ ("#chart").ejChart({ ing hart").ejChart({ the subTitleRendering: () { Chart }}}; sub title. </pre>	<p>Not applicable</p>
		<pre> Fires before Property:toolTipInitialize\$ ("#chart").ejChart({ render rt").ejChart({ </pre>	<p>Property:tooltipRender</p> <pre> er let chart: Chart = new </pre>

		<p>ing the tooltip .</p> <pre> tooltipInitialize: () { Chart({ tooltipRender : () => { }});chart.append To('#chart'); } } }); } } . </pre> <p>Fires before rendering crosshair tooltip in axis</p> <pre> Property:trackAxisToolTip\$("#chart") .ejChart({ trackAxisToolTip: () { } }); } } . </pre> <p>Fires before rendering trackball tooltip .</p> <pre> Property:trackToolTip\$("#chart") .ejChart({ trackToolTip: () { }}); } } . </pre> <p>Event triggered when scroll starts.</p> <pre> Property:scrollStart\$("#chart") = new Chart({ .ejChart({ scrollStart: () { }}); => { }});chart.append To('#chart'); } } } . </pre> <p>Event triggered when scroll ends.</p> <pre> Property:scrollEnd\$("#chart") .ejChart({ scrollEnd: () { }}); } } . </pre> <p>Event triggered when scroll changes.</p> <pre> Property:scrollChange\$("#chart") .ejChart({ scrollChange: () { }}); } } . </pre> <p>Fires while performing</p> <pre> Property:zoomComplete\$("#chart") .ejChart({ zoomComplete: () { }}); } } . </pre>	<p>Chart({ tooltipRender : () => { }});chart.append To('#chart');</p> <p>Not applicable</p> <p>Not applicable</p> <p>Property:scrollStart let chart: Chart = new Chart({ scrollStart: () => { }});chart.append To('#chart');</p> <p>Property:scrollEnd let chart: Chart = new Chart({ scrollEnd: () => { }});chart.append To('#chart');</p> <p>Property:scrollChange let chart: Chart = new Chart({ scrollChange: () => { }});chart.append To('#chart');</p> <p>Property:zoomComplete let chart: Chart = new Chart({</p>
--	--	---	--

		<p>ming rectan gle zoomi ng in chart.</p> <p>Behaviour</p> <p>selected data index</p> <p>sideBySid eSeriesPla cement for column based series</p> <p>ZoomSetti ngs</p> <p>Backgrou nd color of the chart</p> <p>URL of the image to be used as</p>	<pre> zoomComplete: () => { });chart.append To('#chart'); ## Chart properties API in Essential JS 1 Property:selectedDataP ointIndexes:\$("#chart ").ejChart({ selectedDataPointI ndexes: [{ seriesIndex: 0, pointIndex: 1}]); API in Essential JS 2 Property:selectedDataIndex eslet chart: Chart = new Chart({selectedDataIn dexes: [{ series: 0, point: 1}]);chart.appendTo('#chart'); Property:sideBySideSeri esPlacement:\$("#char t").ejChart({ sideBySideSeriesPl acement}); Property:sideBySidePlaceme ntlet chart: Chart = new Chart({ sideBySidePlacement: true});chart.appendTo ('#chart'); Property:zoomSettingslet chart: Chart = new Chart({ zoomSettings: { enable: true, enablePinchZooming: true, enableDeferredZoom : true, enablePinch: true, enableMouseWheel: true, enableScrollBar: true, toolBarItems: [], type: 'X' })); Property:backgroundlet chart: Chart = new Chart({ background: '#EEFFCC'});chart.app endTo('#chart'); Property:backGroundIm ageUrl \$("#container").ej Chart({ backGroundImageUrl </pre>
--	--	---	--

		<pre> chart : background. ../images/chart/w heat.png'}}); </pre>
Customizing border of the chart	<pre> Property: border \$("#container").ej Chart({ border: { width: 2, color: '#CCEEFF', opacity: 0.5}}); </pre>	<pre> Property: border let chart: Chart = new Chart({ border: { width: 2, color: '#CCEEFF'}});chart.ap pendTo('#chart'); </pre>
This provides options for customizing export settings	<pre> Property: exportSettings \$("#container").ej Chart({ exportSettings: { filename : "chart", angle: '45' }}); </pre>	<pre> Property: export() let chart: Chart = new Chart({ border: { width: 2, color: '#CCEEFF'}});chart.ap pendTo('#chart');char t.export(type, fileName); </pre>
ChartArea customization	<pre> Property: chartArea \$("#container").ejChar t({ chartArea: { background: 'transparent', border: { opacity: 0.3, color: 'red', width: 2}}}); </pre>	<pre> Property: chartArea let chart: Chart = new Chart({ chartArea: { background: 'transparent', border: { width: 2, color: '#CCEEFF' }});chart.appendTo('# chart');chart.export(type, fileName); </pre>

primaryYAxis

<!-- markdownlint-disable MD033 -->

Behaviour	API in Essential JS 1	API in Essential JS 2
Alternate grid band	<pre> Property: alternateGridBand \$("#container").ejChart({ primaryYAxis: { alternateGridBand: { even: { fill: 'red' }}}}); </pre>	Not applicable
Axis line cross value	<pre> Property: crossesAt \$("#container").ejChart({ primaryYAxis: { crossesAt: 0 }}); </pre>	<pre> Property: crossesAt let chart: Chart = new Chart({ primaryYAxis: { crossesAt: 4}});chart.appendTo('#ch art'); </pre>
axis name with which the axis line has to be crossed	<pre> Property: crossesInAxis \$("#container").ejChart({ primaryYAxis: { crossesInAxis: '' }}); </pre>	<pre> Property: crossesInAxis let chart: Chart = new Chart({ primaryYAxis: { crossesInAxis: '' </pre>

		<code>}});chart.appendTo('#chart');</code>
axis elements placed with axis line	Property:showNextToAxisLine <code>\$("#container").ejChart({ primaryYAxis: { showNextToAxisLine : true }});</code>	Property:placeNextToAxisLine <code>let chart: Chart = new Chart({ primaryYAxis: { placeNextToAxisLine: ' ' }});chart.appendTo('#chart');</code>
axis line style	Property:axisLine.color <code>\$("#container").ejChart({ primaryYAxis: { axisLine: { color : 'red' } }});</code>	Property:lineStyle.color <code>let chart: Chart = new Chart({ primaryYAxis: { lineStyle: { color: 'black' } }});chart.appendTo('#chart');</code>
axis line dashArray	Property:axisLine.dashArray <code>\$("#container").ejChart({ primaryYAxis: { axisLine: { dashArray : '10, 5' } }});</code>	Property:lineStyle.dashArray <code>let chart: Chart = new Chart({ primaryYAxis: { lineStyle: { dashArray: '10, 5' } }});chart.appendTo('#chart');</code>
Offset for axis	Property:axisLine.offset <code>\$("#container").ejChart({ primaryYAxis: { axisLine: { offset : 10 } }});</code>	Property:plotOffset <code>let chart: Chart = new Chart({ primaryYAxis: { plotOffset: 10 } }});chart.appendTo('#chart');</code>
Visible of an axis	Property:axisLine.visible <code>\$("#container").ejChart({ primaryYAxis: { axisLine: { visible : false } }});</code>	Property:visible <code>let chart: Chart = new Chart({ primaryYAxis: { visible: false } }});chart.appendTo('#chart');</code>
Width of an axis	Property:axisLine.width <code>\$("#container").ejChart({ primaryYAxis: { axisLine: { width : 2 } }});</code>	Property:lineStyle.width <code>let chart: Chart = new Chart({ primaryYAxis: { lineStyle: { width: 3 } }});chart.appendTo('#chart');</code>
Column index of an axis	Property:columnIndex <code>\$("#container").ejChart({ primaryYAxis: { columnIndex: 2 } });</code>	Property:columnIndex <code>let chart: Chart = new Chart({ primaryYAxis: { columnIndex: 2 } }});chart.appendTo('#chart');</code>
span of an axis to place	Property:columnSpan <code>\$("#container").ejChart({ primaryYAxis: { columnIndex: 2 } });</code>	Property:span <code>let chart: Chart = new Chart({ primaryYAxis: { span: 2</code>

horizontally or vertically		<code>}});chart.appendTo('#chart');</code>
Crosshair label of an axis	<code>Property:crossHairLabel.visible\$("#container").ejChart({ primaryYAxis: { crossHairLabel: { visible: true }}});</code>	<code>Property:crossHairTooltip.enable let chart: Chart = new Chart({ primaryYAxis: { crossHairTooltip: { enable: true }}});chart.appendTo('#chart');</code>
Crosshair label color of an axis	Not applicable	<code>Property:crossHairTooltip.fill let chart: Chart = new Chart({ primaryYAxis: { crossHairTooltip: { fill: 'red' }}});chart.appendTo('#chart');</code>
Crosshair label text style	Not applicable	<code>Property:crossHairTooltip.textStyle let chart: Chart = new Chart({ primaryYAxis: { crossHairTooltip: { textStyle: { } }}});chart.appendTo('#chart');</code>
Desired interval count for primaryYAxis	<code>Property:desiredIntervals\$("#container").ejChart({ primaryYAxis: { desiredIntervals: 4}});</code>	<code>Property:desiredIntervals let chart: Chart = new Chart({ primaryYAxis: { desiredIntervals: 4 }});chart.appendTo('#chart');</code>
Edges primaryYAxis	<code>Property:edgeLabelPlacement\$("#container").ejChart({ primaryYAxis: { edgeLabelPlacement: 'none' }});</code>	<code>Property:edgeLabelPlacement let chart: Chart = new Chart({ primaryYAxis: { edgeLabelPlacement: 'Shift' }});chart.appendTo('#chart');</code>
Enables trim for axis labels	<code>Property:enableTrim\$("#container").ejChart({ primaryYAxis: { enableTrim: true }});</code>	<code>Property:enableTrim let chart: Chart = new Chart({ primaryYAxis: { enableTrim: true }});chart.appendTo('#chart');</code>
Specifies the interval of the axis according to the zoomed data of the chart	<code>Property:enableAutoIntervalOnZooming\$("#container").ejChart({ primaryYAxis: { enableAutoIntervalOnZooming: true }});</code>	<code>Property:enableAutoIntervalOnZooming let chart: Chart = new Chart({ primaryYAxis: { enableAutoIntervalOnZooming: true }}</code>

		<code>}});chart.appendTo('#chart');</code>
Specifies the interval of the axis according to the zoomed data of the chart	<code>Property:enableAutoIntervalOnZooming\$("#container").ejChart({ primaryYAxis: { enableAutoIntervalOnZooming: true }});</code>	<code>Property:enableAutoIntervalOnZoominglet chart: Chart = new Chart({ primaryYAxis: { enableAutoIntervalOnZooming: true }});chart.appendTo('#chart');</code>
Font style for primaryYAxis	<code>Property:font\$("#container").ejChart({ primaryYAxis: { font: { fontFamily: 'Calibri', fontStyle: 'italic', fontWeight: '', opacity: 0.5, size: 12 } }});</code>	<code>Property:titleStylelet chart: Chart = new Chart({ primaryYAxis: { titleStyle: { } }});chart.appendTo('#chart');</code>
Indexed for category axis	<code>Property:isIndexed\$("#container").ejChart({ primaryYAxis: { isIndexed: true }});</code>	<code>Property:isIndexedlet chart: Chart = new Chart({ primaryYAxis: { isIndexed: true }});chart.appendTo('#chart');</code>
Interval type for date time axis	<code>Property:intervalType\$("#container").ejChart({ primaryYAxis: { intervalType: 'Auto' }});</code>	<code>Property:intervalTypelet chart: Chart = new Chart({ primaryYAxis: { intervalType: 'Auto' }});chart.appendTo('#chart');</code>
Inversed axis	<code>Property:isInversed\$("#container").ejChart({ primaryYAxis: { isInversed: true }});</code>	<code>Property:isInversedlet chart: Chart = new Chart({ primaryYAxis: { isInversed: true }});chart.appendTo('#chart');</code>
Custom label format	<code>Property:labelFormat\$("#container").ejChart({ primaryYAxis: { labelFormat: '{value}K' }});</code>	<code>Property:labelFormatlet chart: Chart = new Chart({ primaryYAxis: { labelFormat: '{value}K' }});chart.appendTo('#chart');</code>
labelIntersect Action	<code>Property:labelIntersectAction\$("#container").ejChart({ primaryYAxis: { labelIntersectAction: 'trim' }});</code>	<code>Property:labelIntersectActionlet chart: Chart = new Chart({ primaryYAxis: { labelIntersectAction: 'Trim' }});chart.appendTo('#chart');</code>

labelPosition	Property:labelPosition\$("#container").ejChart({ primaryYAxis: { labelPosition: 'inside' } });	Property:labelPositionlet chart: Chart = new Chart({ primaryYAxis: { labelPosition: 'Inside' } });chart.appendTo('#chart');
labelPlacement for category axis	Property:labelPlacement\$("#container").ejChart({ primaryYAxis: { labelPlacement: 'onTicks' } });	Property:labelPlacementlet chart: Chart = new Chart({ primaryYAxis: { labelPlacement: 'OnTicks' } });chart.appendTo('#chart');
Axis label alignment	Property:alignment\$("#container").ejChart({ primaryYAxis: { alignment: 'center' } });	Not Applicable
Rotation of axis labels	Property:labelRotation\$("#container").ejChart({ primaryYAxis: { labelRotation: 45 } });	Property:labelRotationlet chart: Chart = new Chart({ primaryYAxis: { labelRotation: 45 } });chart.appendTo('#chart');
Log base value for logarithmic axis	Property:logBase\$("#container").ejChart({ primaryYAxis: { logBase: 10 } });	Property:labelRotationlet chart: Chart = new Chart({ primaryYAxis: { logBase: 10 } });chart.appendTo('#chart');
Major grid line	Property:majorGridLines.visible\$("#container").ejChart({ primaryYAxis: { majorGridLines: { visible: true } } });	Not Applicable
Width of MajorGridLines	Property:majorGridLines.width\$("#container").ejChart({ primaryYAxis: { majorGridLines: { width: 2 } } });	Property:majorGridLines.width1let chart: Chart = new Chart({ primaryYAxis: { majorGridLines: { width: 2 } } });chart.appendTo('#chart');
Color of MajorGridLines	Property:majorGridLines.color\$("#container").ejChart({ primaryYAxis: { majorGridLines: { color: 'black' } } });	Property:majorGridLines.colorlet chart: Chart = new Chart({ primaryYAxis: { majorGridLines: { color: 'black' } } });chart.appendTo('#chart');
DashArray of MajorGridLines	Property:majorGridLines.dashArray\$("#container").ejChart({ primaryYAxis: { majorGridLines: { dashArray: 'black' } } });	Property:majorGridLines.dashArraylet chart: Chart = new Chart({ primaryYAxis: { majorGridLines: { dashArray: 'black' } } });

		<code>}});chart.appendTo('#chart');</code>
Opacity of major grid line	<code>Property:majorGridLines.opacity\$("#container").ejChart({ primaryYAxis: { majorGridLines: { opacity: true} }});</code>	Not Applicable
Major Tick line	<code>Property:majorTickLines.visible\$("#container").ejChart({ primaryYAxis: { majorTickLines: { visible: true} }});</code>	Not Applicable
Width of MajorTickLines	<code>Property:majorTickLines.width\$("#container").ejChart({ primaryYAxis: { majorTickLines: { width: 2} }});</code>	<code>Property:majorTickLines.width1</code> <code>let chart: Chart = new Chart({ primaryYAxis: { majorTickLines: { width: 2} }});chart.appendTo('#chart');</code>
Height of MajorTickLines	<code>Property:majorTickLines.size\$("#container").ejChart({ primaryYAxis: { majorTickLines: { size: 2} }});</code>	<code>Property:majorTickLines.height1</code> <code>let chart: Chart = new Chart({ primaryYAxis: { majorTickLines: { height: 2} }});chart.appendTo('#chart');</code>
Color of MajorTickLines	<code>Property:majorTickLines.color\$("#container").ejChart({ primaryYAxis: { majorTickLines: { color: 'black' } }});</code>	<code>Property:majorTickLines.color1</code> <code>let chart: Chart = new Chart({ primaryYAxis: { majorTickLines: { color: 'black' } }});chart.appendTo('#chart');</code>
Opacity of major Tick line	<code>Property:majorTickLines.opacity\$("#container").ejChart({ primaryYAxis: { majorTickLines: { opacity: true} }});</code>	Not Applicable
maximum labels of primaryYAxis	<code>Property:maximumLabels\$("#container").ejChart({ primaryYAxis: { maximumLabels: 5 } }});</code>	<code>Property:maximumLabels1</code> <code>let chart: Chart = new Chart({ primaryYAxis: { maximumLabels: 4 } }});chart.appendTo('#chart');</code>
maximum labels width of primaryYAxis to trim	<code>Property:maximumLabelWidth\$("#container").ejChart({ primaryYAxis: { maximumLabelWidth: 40 } }});</code>	<code>Property:maximumLabelWidth1</code> <code>let chart: Chart = new Chart({ primaryYAxis: { maximumLabelWidth: 4 } }});chart.appendTo('#chart');</code>
minor grid line	<code>Property:minorGridLines.visible\$("#container").ejChart({ primaryYAxis: { minorGridLines: { visible: true} }});</code>	Not Applicable

Width of minorGridLines	Property:minorGridLines.width\$("#container").ejChart({ primaryYAxis: { minorGridLines: { width: 2} } });	Property:minorGridLines.width let chart: Chart = new Chart({ primaryYAxis: { minorGridLines: { width: 2} } });chart.appendTo('#chart');
Color of minorGridLines	Property:minorGridLines.color\$("#container").ejChart({ primaryYAxis: { minorGridLines: { color: 'black' } } });	Property:minorGridLines.color let chart: Chart = new Chart({ primaryYAxis: { minorGridLines: { color: 'black' } } });chart.appendTo('#chart');
DashArray of minorGridLines	Property:minorGridLines.dashArray\$("#container").ejChart({ primaryYAxis: { minorGridLines: { dashArray: 'black' } } });	Property:minorGridLines.dashArray let chart: Chart = new Chart({ primaryYAxis: { minorGridLines: { dashArray: 'black' } } });chart.appendTo('#chart');
Opacity of minor grid line	Property:minorGridLines.opacity\$("#container").ejChart({ primaryYAxis: { minorGridLines: { opacity: true} } });	Not Applicable
minor Tick line	Property:minorTickLines.visible\$("#container").ejChart({ primaryYAxis: { minorTickLines: { visible: true} } });	Not Applicable
Width of minorTickLines	Property:minorTickLines.width\$("#container").ejChart({ primaryYAxis: { minorTickLines: { width: 2} } });	Property:minorTickLines.width let chart: Chart = new Chart({ primaryYAxis: { minorTickLines: { width: 2} } });chart.appendTo('#chart');
Height of minorTickLines	Property:minorTickLines.size\$("#container").ejChart({ primaryYAxis: { minorTickLines: { size: 2} } });	Property:minorTickLines.height let chart: Chart = new Chart({ primaryYAxis: { minorTickLines: { height: 2} } });chart.appendTo('#chart');
Color of minorTickLines	Property:minorTickLines.color\$("#container").ejChart({ primaryYAxis: { minorTickLines: { color: 'black' } } });	Property:minorTickLines.color let chart: Chart = new Chart({ primaryYAxis: { minorTickLines: { color: 'black' } } });chart.appendTo('#chart');

Opacity of minor Tick line	Property:minorTickLines.opacity\$("#container").ejChart({ primaryYAxis: { minorTickLines: { opacity: true} }});	Not Applicable
Minor ticks per interval of primaryYAxis	Property:minorTicksPerInterval\$("#container").ejChart({ primaryYAxis: { minorTicksPerInterval: 4 }});	Property:minorTickLines.colorlet chart: Chart = new Chart({ primaryYAxis: { minorTicksPerInterval: 4 }});chart.appendTo('#chart');
name of the primaryYAxis	Property:name\$("#container").ejChart({ primaryYAxis: { name: 'primaryYAxis' }});	Property:namelet chart: Chart = new Chart({ primaryYAxis: { name: 'primaryYAxis' }});chart.appendTo('#chart');
Orientation of primaryYAxis	Property:orientation\$("#container").ejChart({ primaryYAxis: { orientation: 'Vertical' }});	Not Applicable
Plot offset for primaryYAxis	Property:plotOffset\$("#container").ejChart({ primaryYAxis: { plotOffset: 0 }});	Property:plotOffsetlet chart: Chart = new Chart({ primaryYAxis: { plotOffset: 0 }});chart.appendTo('#chart');
minimum for primaryYAxis	Property:range.minimum\$("#container").ejChart({ primaryYAxis: { range: { minimum: 10 } }});	Property:minimumlet chart: Chart = new Chart({ primaryYAxis: { minimum: 23 }});chart.appendTo('#chart');
maximum for primaryYAxis	Property:range.maximum\$("#container").ejChart({ primaryYAxis: { range: { maximum: 10 } }});	Property:maximumlet chart: Chart = new Chart({ primaryYAxis: { maximum: 23 }});chart.appendTo('#chart');
interval for primaryYAxis	Property:range.interval\$("#container").ejChart({ primaryYAxis: { range: { interval: 1 } }});	Property:intervallet chart: Chart = new Chart({ primaryYAxis: { interval: 2 }});chart.appendTo('#chart');
RangePadding for primaryYAxis	Property:rangePadding\$("#container").ejChart({ primaryYAxis: { rangePadding: 'None' }});	Property:rangePaddinglet chart: Chart = new Chart({ primaryYAxis: { rangePadding: 'None' }});chart.appendTo('#chart');

Rounding Places in primaryYAxis	Property:roundingPlaces \$("#container").ejChart({ primaryYAxis: { roundingPlaces: 3 } });	Property:labelFormatlet chart: Chart = new Chart({ primaryYAxis: { labelFormat: 'n3' }});chart.appendTo('#cha rt');
ScrollBar settings of primaryYAxis	Property:scrollbarSettings \$("#container").ejChart({ primaryYAxis: { scrollbarSettings : { } } });	Not Applicable
TickPosition in primaryYAxis	Property:tickLinesPosition\$("#container").ejCha rt({ primaryYAxis: { tickLinesPosition: 'Inside' } });	Property:tickPositionlet chart: Chart = new Chart({ primaryYAxis: { tickPosition: 'Inside' }});chart.appendTo('#cha rt');
valueType of primaryYAxis	Property:valueType\$("#container").ejChart({ primaryYAxis: { valueType: 'DateTime' }});	Property:valueTypelet chart: Chart = new Chart({ primaryYAxis: { valueType: 'DateTime' }});chart.appendTo('#cha rt');
visible of primaryYAxis	Property:visible\$("#container").ejChart({ primaryYAxis: { visible: true } });	Property:visiblelet chart: Chart = new Chart({ primaryYAxis: { visible: true }});chart.appendTo('#cha rt');
zoomFactor of primaryYAxis	Property:zoomFactor\$("#container").ejChart({ primaryYAxis: { zoomFactor: 0.3 } });	Property:zoomFactorlet chart: Chart = new Chart({ primaryYAxis: { zoomFactor: 0.3 }});chart.appendTo('#cha rt');
zoomPosition of primaryYAxis	Property:zoomPosition\$("#container").ejChart({ primaryYAxis: { zoomPosition: 0.3 } });	Property:zoomPositionlet chart: Chart = new Chart({ primaryYAxis: { zoomPosition: 0.3 }});chart.appendTo('#cha rt');
labelBorder of primaryYAxis	Property:labelBorder\$("#container").ejChart({ primaryYAxis: { labelBorder: { color: 'red', width: 2 } } });	Property:borderlet chart: Chart = new Chart({ primaryYAxis: { border: { color: 'red', width: 3 } }});chart.appendTo('#cha rt');

title of primaryYAxis	Property:title.text <code>\$("#container").ejChart({ primaryYAxis: { title: { text: 'Chart title' } } });</code>	Property:title <code>let chart: Chart = new Chart({ primaryYAxis: { title: 'Chart title' } });chart.appendTo('#chart');</code>
StripLine of primaryYAxis	Property:stripLine <code>\$("#container").ejChart({ primaryYAxis: { stripLine: [] } });</code>	Property:stripLines <code>let chart: Chart = new Chart({ primaryYAxis: { stripLines: [] } });chart.appendTo('#chart');</code>
Multilevel labels of primaryYAxis	Property:multiLevelLabels <code>\$("#container").ejChart({ primaryYAxis: { multiLevelLabels: [] } });</code>	Property:multiLevelLabels <code>let chart: Chart = new Chart({ primaryYAxis: { stripLines: [] } });chart.appendTo('#chart');</code>
skeleton for an axes	Not Applicable	Property:skeleton <code>let chart: Chart = new Chart({ axes: [{ skeleton: 'yMd' }] });chart.appendTo('#chart');</code>
skeleton type for an axes	Not Applicable	Property:skeletonType <code>let chart: Chart = new Chart({ axes: [{ skeletonType: 'DateTime' }] });chart.appendTo('#chart');</code>

Axes

<!-- markdownlint-disable MD033 -->

Behaviour	API in Essential JS 1	API in Essential JS 2
Alternate grid band	Property:alternateGridBand <code>\$("#container").ejChart({ axes: [{ alternateGridBand: { even: { fill: 'red' } } }] });</code>	Not applicable
Axis line cross value	Property:crossesAt <code>\$("#container").ejChart({ axes: [{ crossesAt: 0 }] });</code>	Property:crossesAt <code>let chart: Chart = new Chart({ axes: [{ crossesAt: 4 }] });chart.appendTo('#chart');</code>

axis name with which the axis line has to be crossed	Property:crossesInAxis\$("#container").ejChart({ axes: [{ crossesInAxis: ' ' }] });	Property:crossesInAxislet chart: Chart = new Chart({ axes: [{ crossesInAxis: ' ' }] });chart.appendTo('#chart');
axis elements placed with axis line	Property:showNextToAxisLine\$("#container").ejChart({ axes: [{ showNextToAxisLine : true }] });	Property:placeNextToAxisLinelet chart: Chart = new Chart({ axes: [{ placeNextToAxisLine: ' ' }] });chart.appendTo('#chart');
axis line style	Property:axisLine.color\$("#container").ejChart({ axes: [{ axisLine: { color : 'red' } }] });	Property:lineStyle.colorlet chart: Chart = new Chart({ axes: [{ lineStyle: { color: 'black' } }] });chart.appendTo('#chart');
axis line dashArray	Property:axisLine.color\$("#container").ejChart({ axes: [{ axisLine: { dashArray : '10, 5' } }] });	Property:lineStyle.dashArraylet chart: Chart = new Chart({ axes: [{ lineStyle: { dashArray: '10, 5' } }] });chart.appendTo('#chart');
Offset for axis	Property:axisLine.offset\$("#container").ejChart({ axes: [{ axisLine: { offset : 10 } }] });	Property:plotOffsetlet chart: Chart = new Chart({ axes: [{ plotOffset: 10 }] });chart.appendTo('#chart');
Visible of an axis	Property:axisLine.offset\$("#container").ejChart({ axes: [{ axisLine: { visible : false } }] });	Property:visiblelet chart: Chart = new Chart({ axes: [{ visible: false }] });chart.appendTo('#chart');
Width of an axis	Property:axisLine.width\$("#container").ejChart({ axes: [{ axisLine: { width : 2 } }] });	Property:lineStyle.widthlet chart: Chart = new Chart({ axes: [{ lineStyle: { width: 3 } }] });chart.appendTo('#chart');
Column index of an axis	Property:columnIndex\$("#container").ejChart({ axes: [{ columnIndex: 2 }] });	Property:columnIndexlet chart: Chart = new Chart({ axes: [{ columnIndex: 2 }] });chart.appendTo('#chart');
span of an axis to place	Property:columnSpan\$("#container").ejChart({ axes: [{ columnIndex: 2 }] });	Property:spanlet chart: Chart = new Chart({ axes: [{ span: 2 }] });chart.appendTo('#chart');

horizontally or vertically		
Crosshair label of an axis	Property:crossHairLabel.visible\$("#container").ejChart({ axes: [{ crossHairLabel: { visible: true }}}]);	Property:crossHairTooltip.enablelet chart: Chart = new Chart({ axes: [{ crossHairTooltip: { enable: true }}}]);chart.appendTo('#chart');
Crosshair label color of an axis	Not applicable	Property:crossHairTooltip.filllet chart: Chart = new Chart({ axes: [{ crossHairTooltip: { fill: 'red' }}}]);chart.appendTo('#chart');
Crosshair label text style	Not applicable	Property:crossHairTooltip.textStylelet chart: Chart = new Chart({ axes: [{ crossHairTooltip: { textStyle: { } }}}]);chart.appendTo('#chart');
Desired interval count for primary YAxis	Property:desiredIntervals\$("#container").ejChart({ axes: [{ desiredIntervals: 4 }}}]);	Property:desiredIntervalslet chart: Chart = new Chart({ axes: [{ desiredIntervals: 4 }}}]);chart.appendTo('#chart');
Edges primary YAxis	Property:edgeLabelPlacement\$("#container").ejChart({ axes: [{ edgeLabelPlacement: 'none' }}}]);	Property:edgeLabelPlacementlet chart: Chart = new Chart({ axes: [{ edgeLabelPlacement: 'Shift' }}}]);chart.appendTo('#chart');
Enables trim for axis labels	Property:enableTrim\$("#container").ejChart({ axes: [{ enableTrim: true }}}]);	Property:enableTrimlet chart: Chart = new Chart({ axes: [{ enableTrim: true }}}]);chart.appendTo('#chart');
Specifies the interval of the axis according to the zoomed data	Property:enableAutoIntervalOnZooming\$("#container").ejChart({ axes: [{ enableAutoIntervalOnZooming: true }}}]);	Property:enableAutoIntervalOnZoominglet chart: Chart = new Chart({ axes: [{ enableAutoIntervalOnZooming: true }}}]);chart.appendTo('#chart');

of the chart		
Specifies the interval of the axis according to the zoomed data of the chart	Property:enableAutoIntervalOnZooming\$("#container").ejChart({ axes: [{ enableAutoIntervalOnZooming: true }]});	Property:enableAutoIntervalOnZoominglet chart: Chart = new Chart({ axes: [{ enableAutoIntervalOnZooming: true }]});chart.appendTo('#chart');
Font style for primary YAxis	Property:font\$("#container").ejChart({ axes: [{ font: { fontFamily: 'Calibri', fontStyle: 'italic', fontWeight: '', opacity: 0.5, size: 12 } }]});	Property:titleStylelet chart: Chart = new Chart({ axes: [{ titleStyle: { } }]});chart.appendTo('#chart');
Indexed for category axis	Property:isIndexed\$("#container").ejChart({ axes: [{ isIndexed: true }]});	Property:isIndexedlet chart: Chart = new Chart({ axes: [{ isIndexed: true }]});chart.appendTo('#chart');
Interval type for date time axis	Property:intervalType\$("#container").ejChart({ axes: [{ intervalType: 'Auto' }]});	Property:intervalTypelet chart: Chart = new Chart({ axes: [{ intervalType: 'Auto' }]});chart.appendTo('#chart');
Inversed axis	Property:isInversed\$("#container").ejChart({ axes: [{ isInversed: true }]});	Property:isInversedlet chart: Chart = new Chart({ axes: [{ isInversed: true }]});chart.appendTo('#chart');
Custom label format	Property:labelFormat\$("#container").ejChart({ axes: [{ labelFormat: '{value}K' }]});	Property:labelFormatlet chart: Chart = new Chart({ axes: [{ labelFormat: '{value}K' }]});chart.appendTo('#chart');
labelIntersection	Property:labelIntersection\$("#container").ejChart({ axes: [{	Property:labelIntersectionlet chart: Chart = new Chart({ axes: [{ labelIntersection: 'Trim' }]});chart.appendTo('#chart');

	<code>labelIntersectAction: 'trim' }]]]);</code>	
labelPosition	<code>Property:labelPosition\$("#container").ejChart({ axes: [{ labelPosition: 'inside' }]]);</code>	<code>Property:labelPositionlet chart: Chart = new Chart({ axes: [{ labelPosition: 'Inside' }]});chart.appendTo('#chart');</code>
labelPlacement for category axis	<code>Property:labelPlacement\$("#container").ejChart({ axes: [{ labelPlacement: 'onTicks' }]]);</code>	<code>Property:labelPlacementlet chart: Chart = new Chart({ axes: [{ labelPlacement: 'OnTicks' }]});chart.appendTo('#chart');</code>
Axis label alignment	<code>Property:alignment\$("#container").ejChart({ axes: [{ alignment: 'center' }]]);</code>	Not Applicable
Rotation of axis labels	<code>Property:labelRotation\$("#container").ejChart({ axes: [{ labelRotation: 45 }]]);</code>	<code>Property:labelRotationlet chart: Chart = new Chart({ axes: [{ labelRotation: 45 }]});chart.appendTo('#chart');</code>
Log base value for logarithmic axis	<code>Property:logBase\$("#container").ejChart({ axes: [{ logBase: 10 }]]);</code>	<code>Property:labelRotationlet chart: Chart = new Chart({ axes: [{ logBase: 10 }]});chart.appendTo('#chart');</code>
Major grid line	<code>Property:majorGridLines.visible\$("#container").ejChart({ axes: [{ majorGridLines: { visible: true } }]]);</code>	Not Applicable
Width of MajorGridLines	<code>Property:majorGridLines.width\$("#container").ejChart({ axes: [{ majorGridLines: { width: 2 } }]]);</code>	<code>Property:majorGridLines.widthlet chart: Chart = new Chart({ axes: [{ majorGridLines: { width: 2 } }]});chart.appendTo('#chart');</code>
Color of MajorGridLines	<code>Property:majorGridLines.color\$("#container").ejChart({ axes: [{ majorGridLines: { color: 'black' } }]]);</code>	<code>Property:majorGridLines.colorlet chart: Chart = new Chart({ axes: [{ majorGridLines: { color: 'black' } }]});chart.appendTo('#chart');</code>

DashArray of MajorGridLines	Property:majorGridLines.dashArray\$("#container").ejChart({ axes: [{ majorGridLines: { dashArray: 'black' } }] });	Property:majorGridLines.dashArraylet chart: Chart = new Chart({ axes: [{ majorGridLines: { dashArray: 'black' } }] });chart.appendTo('#chart');
Opacity of major grid line	Property:majorGridLines.opacity\$("#container").ejChart({ axes: [{ majorGridLines: { opacity: true } }] });	Not Applicable
Major Tick line	Property:majorTickLines.visible\$("#container").ejChart({ axes: [{ majorTickLines: { visible: true } }] });	Not Applicable
Width of MajorTickLines	Property:majorTickLines.width\$("#container").ejChart({ axes: [{ majorTickLines: { width: 2 } }] });	Property:majorTickLines.widthlet chart: Chart = new Chart({ axes: [{ majorTickLines: { width: 2 } }] });chart.appendTo('#chart');
Height of MajorTickLines	Property:majorTickLines.size\$("#container").ejChart({ axes: [{ majorTickLines: { size: 2 } }] });	Property:majorTickLines.heightlet chart: Chart = new Chart({ axes: [{ majorTickLines: { height: 2 } }] });chart.appendTo('#chart');
Color of MajorTickLines	Property:majorTickLines.color\$("#container").ejChart({ axes: [{ majorTickLines: { color: 'black' } }] });	Property:majorTickLines.colorlet chart: Chart = new Chart({ axes: [{ majorTickLines: { color: 'black' } }] });chart.appendTo('#chart');
Opacity of major Tick line	Property:majorTickLines.opacity\$("#container").ejChart({ axes: [{ majorTickLines: { opacity: true } }] });	Not Applicable
maximum labels of primaryYAxis	Property:maximumLabels\$("#container").ejChart({ axes: [{ maximumLabels: 5 }] });	Property:maximumLabelslet chart: Chart = new Chart({ axes: [{ maximumLabels: 4 }] });chart.appendTo('#chart');

maximum labels width of primary YAxis to trim	Property:maximumLabelWidth\$("#container").ejChart({ axes: [{ maximumLabelWidth: 40 }] });	Property:maximumLabelWidthlet chart: Chart = new Chart({ axes: [{ maximumLabelWidth: 40 }] });chart.appendTo('#chart');
minor grid line	Property:minorGridLines.visible\$("#container").ejChart({ axes: [{ minorGridLines: { visible: true } }] });	Not Applicable
Width of minorGridLines	Property:minorGridLines.width\$("#container").ejChart({ axes: [{ minorGridLines: { width: 2 } }] });	Property:minorGridLines.widthlet chart: Chart = new Chart({ axes: [{ minorGridLines: { width: 2 } }] });chart.appendTo('#chart');
Color of minorGridLines	Property:minorGridLines.color\$("#container").ejChart({ axes: [{ minorGridLines: { color: 'black' } }] });	Property:minorGridLines.colorlet chart: Chart = new Chart({ axes: [{ minorGridLines: { color: 'black' } }] });chart.appendTo('#chart');
DashArray of minorGridLines	Property:minorGridLines.dashArray\$("#container").ejChart({ axes: [{ minorGridLines: { dashArray: 'black' } }] });	Property:minorGridLines.dashArraylet chart: Chart = new Chart({ axes: [{ minorGridLines: { dashArray: 'black' } }] });chart.appendTo('#chart');
Opacity of minor grid line	Property:minorGridLines.opacity\$("#container").ejChart({ axes: [{ minorGridLines: { opacity: true } }] });	Not Applicable
minor Tick line	Property:minorTickLines.visible\$("#container").ejChart({ axes: [{ minorTickLines: { visible: true } }] });	Not Applicable
Width of minorTickLines	Property:minorTickLines.width\$("#container").ejChart({ axes: [{ minorTickLines: { width: 2 } }] });	Property:minorTickLines.widthlet chart: Chart = new Chart({ axes: [{ minorTickLines: { width: 2 } }] });chart.appendTo('#chart');

Height of minorTickLines	Property:minorTickLines.size\$("#container").ejChart({ axes: [{ minorTickLines: { size: 2 } }] });	Property:minorTickLines.heightlet chart: Chart = new Chart({ axes: [{ minorTickLines: { height: 2 } }] });chart.appendTo('#chart');
Color of minorTickLines	Property:minorTickLines.color\$("#container").ejChart({ axes: [{ minorTickLines: { color: 'black' } }] });	Property:minorTickLines.colorlet chart: Chart = new Chart({ axes: [{ minorTickLines: { color: 'black' } }] });chart.appendTo('#chart');
Opacity of minorTickLine	Property:minorTickLines.opacity\$("#container").ejChart({ axes: [{ minorTickLines: { opacity: true } }] });	Not Applicable
Minor ticks per interval of primaryYAxis	Property:minorTicksPerInterval\$("#container").ejChart({ axes: [{ minorTicksPerInterval: 4 } }] });	Property:minorTickLines.colorlet chart: Chart = new Chart({ axes: [{ minorTicksPerInterval: 4 } }] });chart.appendTo('#chart');
name of the primaryYAxis	Property:name\$("#container").ejChart({ axes: [{ name: 'primaryYAxis' } }] });	Property:namelet chart: Chart = new Chart({ axes: [{ name: 'primaryYAxis' } }] });chart.appendTo('#chart');
Orientation of primaryYAxis	Property:orientation\$("#container").ejChart({ axes: [{ orientation: 'Vertical' } }] });	Not Applicable
Plot offset for primaryYAxis	Property:plotOffset\$("#container").ejChart({ axes: [{ plotOffset: 0 } }] });	Property:plotOffsetlet chart: Chart = new Chart({ axes: [{ plotOffset: 0 } }] });chart.appendTo('#chart');
minimum for primaryYAxis	Property:range.minimum\$("#container").ejChart({ axes: [{ range: { minimum: 10 } }] });	Property:minimumlet chart: Chart = new Chart({ axes: [{ minimum: 23 } }] });chart.appendTo('#chart');
maximum for	Property:range.maximum\$("#container").ejChar	Property:maximumlet chart: Chart = new Chart({ axes: [{ maximum: 23 } }] });chart.appendTo('#chart');

primaryYAxis	<code>t({ axes: [{ range: { maximum: 10 } }] });</code>	
interval for primaryYAxis	<code>Property:range.interval\$("#container").ejChart({ axes: [{ range: { interval: 1 } }] });</code>	<code>Property:intervallet chart: Chart = new Chart({ axes: [{ interval: 2 }] });chart.appendTo('#chart');</code>
RangePadding for primaryYAxis	<code>Property:rangePadding\$("#container").ejChart({ axes: [{ rangePadding: 'None' }] });</code>	<code>Property:rangePaddinglet chart: Chart = new Chart({ axes: [{ rangePadding: 'None' }] });chart.appendTo('#chart');</code>
Rounding Places in primaryYAxis	<code>Property:roundingPlaces\$("#container").ejChart({ axes: [{ roundingPlaces: 3 }] });</code>	<code>Property:labelFormatlet chart: Chart = new Chart({ axes: [{ labelFormat: 'n3' }] });chart.appendTo('#chart');</code>
Scrollbar settings of primaryYAxis	<code>Property:scrollbarSettings\$("#container").ejChart({ axes: [{ scrollbarSettings: { } }] });</code>	Not Applicable
TickPosition in primaryYAxis	<code>Property:tickLinesPosition\$("#container").ejChart({ axes: [{ tickLinesPosition: 'Inside' }] });</code>	<code>Property:tickPositionlet chart: Chart = new Chart({ axes: [{ tickPosition: 'Inside' }] });chart.appendTo('#chart');</code>
valueType of primaryYAxis	<code>Property:valueType\$("#container").ejChart({ axes: [{ valueType: 'DateTime' }] });</code>	<code>Property:valueTypelet chart: Chart = new Chart({ axes: [{ valueType: 'DateTime' }] });chart.appendTo('#chart');</code>
visible of primaryYAxis	<code>Property:visible\$("#container").ejChart({ axes: [{ visible: true }] });</code>	<code>Property:visiblelet chart: Chart = new Chart({ axes: [{ visible: true }] });chart.appendTo('#chart');</code>
zoomFactor of primaryYAxis	<code>Property:zoomFactor\$("#container").ejChart({ axes: [{ zoomFactor: 0.3 }] });</code>	<code>Property:zoomFactorlet chart: Chart = new Chart({ axes: [{ zoomFactor: 0.3 }] });chart.appendTo('#chart');</code>

zoomPosition of primaryYAxis	Property:zoomPosition\$("#container").ejChart({ axes: [{ zoomPosition: 0.3 }] });	Property:zoomPositionlet chart: Chart = new Chart({ axes: [{ zoomPosition: 0.3 }] });chart.appendTo('#chart');									
labelBorder of primaryYAxis	Property:labelBorder\$("#container").ejChart({ axes: [{ labelBorder: { color: 'red', width: 2 } }] });	Property:borderlet chart: Chart = new Chart({ axes: [{ border: { color: 'red', width: 3 } }] });chart.appendTo('#chart');									
title of primaryYAxis	Property:title.text\$("#container").ejChart({ axes: [{ title: { text: 'Chart title' } }] });	Property:titlelet chart: Chart = new Chart({ axes: [{ title: 'Chart title' }] });chart.appendTo('#chart');									
StripLine of primaryYAxis	Property:stripLine\$("#container").ejChart({ axes: [{ stripLine: [] }] });	Property:stripLineslet chart: Chart = new Chart({ axes: [{ stripLines: [] }] });chart.appendTo('#chart');									
Multilevel labels of axes	Property:multiLevelLabels\$("#container").ejChart({ axes: [{ multiLevelLabels: [] }] });	Property:multiLevelLabelslet chart: Chart = new Chart({ axes: [{ stripLines: [] }] });chart.appendTo('#chart');									
skeleton for an axes	Not Applicable	Property:skeletonlet chart: Chart = new Chart({ axes: [{ skeleton: 'yMd' }] });chart.appendTo('#chart');									
skeleton type for an axes	Not Applicable	Property:skeletonTypelet chart: Chart = new Chart({ axes: [{ skeletonType: 'DateTime' }] });chart.appendTo('#chart'); ## Rows <table> <tr> <td>Behavior</td> <td>API in Essential JS 1</td> <td>API in Essential JS 2</td> </tr> <tr> <td>rows in chart</td> <td>Property:rowDefinitions\$("#chart").ejChart({ rowDefinitions: [] });</td> <td>Property:rowslet chart: Chart = new Chart({ rows: [] });chart.appendTo('#chart');</td> </tr> <tr> <td>unit</td> <td>Property:unit\$("#container").ejChart({ rowDefinitions: [{ unit: "percentage" }] });</td> <td>Not Applicable</td> </tr> </table>	Behavior	API in Essential JS 1	API in Essential JS 2	rows in chart	Property:rowDefinitions\$("#chart").ejChart({ rowDefinitions: [] });	Property:rowslet chart: Chart = new Chart({ rows: [] });chart.appendTo('#chart');	unit	Property:unit\$("#container").ejChart({ rowDefinitions: [{ unit: "percentage" }] });	Not Applicable
Behavior	API in Essential JS 1	API in Essential JS 2									
rows in chart	Property:rowDefinitions\$("#chart").ejChart({ rowDefinitions: [] });	Property:rowslet chart: Chart = new Chart({ rows: [] });chart.appendTo('#chart');									
unit	Property:unit\$("#container").ejChart({ rowDefinitions: [{ unit: "percentage" }] });	Not Applicable									

		<pre> Property:rowHeight\$("#chart").ejChart({ rowDefinitions: [{ rowHeight: '50%'}];}); </pre>	<pre> Property:heightlet chart: Chart = new Chart({ rows: [{ height: '300'}];});chart.appendTo('#chart'); </pre>
	Line custom ization	<pre> Property:lineColor, lineWidth\$("#chart").ejChart({ rowDefinitions: [{ rowHeight: '50%', lineColor: 'brown', lineWidth: 2}];}); </pre>	<pre> Property:borderlet chart: Chart = new Chart({ rows: [{ height: '300', border: { width: 2, color: 'brown'}}];});chart.appendTo('#chart'); </pre>
	## Series		
	Behavior	API in Essential JS 1	API in Essential JS 2
	bearFillColor	<pre> Property:bearFillColor\$("#chart").ejChart({ series: [{bearFillColor: 'red' }];}); </pre>	<pre> Property:bearFillColorlet chart: Chart = new Chart({ series: [{bearFillColor: 'red' }];});chart.appendTo('#chart'); </pre>
	Border	<pre> Property:border\$("#chart").ejChart({ series: [{ border: { color: 'red', width: 2, dashArray: '10, 5' } }];}); </pre>	<pre> Property:rowslet chart: Chart = new Chart({ series: [{ border: { color: 'red', width: 2} }];});chart.appendTo('#chart'); </pre>
	BoxPlot Mode	<pre> Property:boxPlotMode\$("#chart").ejChart({ series: [{ boxPlotMode: 'inclusive' }];}); </pre>	<pre> Property:rowslet chart: Chart = new Chart({ series: [{ boxPlotMode: 'Inclusive' }];});chart.appendTo('#chart'); </pre>
	Minimum radius of Bubble series	<pre> Property:bubbleOptions.minRadius\$("#chart").ejChart({ series: [{ bubbleOptions: { minRadius: 2} }];}); </pre>	<pre> Property:minRadiuslet chart: Chart = new Chart({ series: [{ minRadius: 2 }];});chart.appendTo('#chart'); </pre>

Maximum radius of Bubble series	Property:bubbleOptions.maxRadius\$("#chart").ejChart({ series: [{ bubbleOptions: { maxRadius: 10 } }]});	Property:maxRadius1 let chart: Chart = new Chart({ series: [{ maxRadius: 2 }]});chart.appendTo('#chart');
bullFillColor	Property:bullFillColor\$("#chart").ejChart({ series: [{ bullFillColor: 'red' }]});	Property:bullFillColor let chart: Chart = new Chart({ series: [{ bullFillColor: 'red' }]});chart.appendTo('#chart');
Cardinal spline tension for spline series	Property:cardinalSplineTension\$("#chart").ejChart({ series: [{ cardinalSplineTension: 0.5 }]});	Property:cardinalSplineTension let chart: Chart = new Chart({ series: [{ cardinalSplineTension: 0.5 }]});chart.appendTo('#chart');
Column Width for rectangle series	Property:columnWidth\$("#chart").ejChart({ series: [{ columnWidth: 0.5 }]});	Property:columnWidth let chart: Chart = new Chart({ series: [{ columnWidth: 0.5 }]});chart.appendTo('#chart');
Column spacing for rectangle series	Property:columnSpacing\$("#chart").ejChart({ series: [{ columnSpacing: 0.5 }]});	Property:columnSpacing let chart: Chart = new Chart({ series: [{ columnSpacing: 0.5 }]});chart.appendTo('#chart');
Topleft radius for rectangle series	Property:cornerRadius.topLeft\$("#chart").ejChart({ series: [{ topLeft: 0 }]});	Property:cornerRadius.topLeft let chart: Chart = new Chart({ series: [{ topLeft: 0 }]});chart.appendTo('#chart');

		Property:cornerRadius.topRight Property:cornerRadius.topRight\$ chart: Chart = new Chart({ series: [{ topRight: 0 }];}); chart.appendTo('#chart');	Property:cornerRadius.topRight Property:cornerRadius.topRight\$ chart: Chart = new Chart({ series: [{ topRight: 0 }];}); chart.appendTo('#chart');
		Property:cornerRadius.bottomRight Property:cornerRadius.bottomRight\$ chart: Chart = new Chart({ series: [{ bottomRight: 0 }];}); chart.appendTo('#chart');	Property:cornerRadius.bottomRight Property:cornerRadius.bottomRight\$ chart: Chart = new Chart({ series: [{ bottomRight: 0 }];}); chart.appendTo('#chart');
		Property:cornerRadius.bottomLeft Property:cornerRadius.bottomLeft\$ chart: Chart = new Chart({ series: [{ bottomLeft: 0 }];}); chart.appendTo('#chart');	Property:cornerRadius.bottomLeft Property:cornerRadius.bottomLeft\$ chart: Chart = new Chart({ series: [{ bottomLeft: 0 }];}); chart.appendTo('#chart');
		Property:dashArray Property:dashArray\$ chart: Chart = new Chart({ series: [{ dashArray: '10, 5' }];}); chart.appendTo('#chart');	Property:dashArray Property:dashArray\$ chart: Chart = new Chart({ series: [{ dashArray: '10, 5' }];}); chart.appendTo('#chart');
		Property:dataSource Property:dataSource\$ chart: Chart = new Chart({ series: [{ dataSource: [] }];}); chart.appendTo('#chart');	Property:dataSource Property:dataSource\$ chart: Chart = new Chart({ series: [{ dataSource: [] }];}); chart.appendTo('#chart');
		Property:drawType Property:drawType\$ chart: Chart = new Chart({ series: [{ drawType: 'Line' }];}); chart.appendTo('#chart');	Property:drawType Property:drawType\$ chart: Chart = new Chart({ series: [{ drawType: 'Line' }];}); chart.appendTo('#chart');
		Property:emptyPointSettings.visible Property:emptyPointSettings.visible\$ chart: Chart = new Chart({ series: [{ emptyPointSettings: { visible: false } }];}); chart.appendTo('#chart');	Property:emptyPointSettings.visible Property:emptyPointSettings.visible\$ chart: Chart = new Chart({ series: [{ emptyPointSettings: { visible: false } }];}); chart.appendTo('#chart');

		gs for series <pre>emptyPointSettings: { visible: false } }];});</pre>	
	Empty Point Display mode	<pre>Property:emptyPointSettings.displayMode\$("#chart").ejChart({ series: [{ displayMode: 'gap' }];});</pre>	Property:emptyPointSettings.displayMode <pre>let chart: Chart = new Chart({ series: [{ displayMode: 'Average' }];});chart.appendTo('#chart');</pre>
	Empty Point color	<pre>Property:emptyPointSettings.color\$("#chart").ejChart({ series: [{ color: 'red' }];});</pre>	Property:emptyPointSettings.fill <pre>let chart: Chart = new Chart({ series: [{ fill: 'red' }];});chart.appendTo('#chart');</pre>
	Empty Point Border	<pre>Property:emptyPointSettings.border\$("#chart").ejChart({ series: [{ emptyPointSettings: { color: 'red', width: 2 } }];});</pre>	Property:fill <pre>let chart: Chart = new Chart({ series: [{ emptyPointSettings: { color: 'red', width: 2 }];});chart.appendTo('#chart');</pre>
	Enable animation for series	<pre>Property:enableAnimation\$("#chart").ejChart({ series: [{ enableAnimation: true }];});</pre>	Property:animation.enable <pre>let chart: Chart = new Chart({ series: [animation: { enable: false }]});chart.appendTo('#chart');</pre>
	Animation duration for series	<pre>Property:animationDuration\$("#chart").ejChart({ series: [{ animationDuration: 1000 }];});</pre>	Property:animation.duration <pre>let chart: Chart = new Chart({ series: [animation: { duration: 1000 }]});chart.appendTo('#chart');</pre>
	Animation delay for series	Not Applicable	Property:animation.duration <pre>let chart: Chart = new Chart({ series: [animation: {</pre>

		<pre> delay: 100 });});chart.appe ndTo('#chart'); </pre>
Drag settings for series	<pre> Property:dragSettings\$("#char t").ejChart({ series: [{ dragSettings: { mode: 'X' } }]);}); </pre>	Not Applicable
Errorbar settings for series	<pre> Property:errorBarSettings\$("#c hart").ejChart({ series: [{ errorBarSettings: { } }]);}); </pre>	<pre> Property:errorBarSett ingslet chart: Chart = new Chart({ series: [{errorBarSetting s: { } }]);}); </pre>
Closed series	<pre> Property:isClosed\$("#chart") .ejChart({ series: [{ isClosed: true }]);}); </pre>	<pre> Property:isClosedlet chart: Chart = new Chart({ series: [{ isClosed: true }]);}); </pre>
Stacking Property for series	<pre> Property:isStacking\$("#chart").ejChart({ series: [{ isStacking: true }]);}); </pre>	Not Applicable
Line cap for series	<pre> Property:lineCap\$("#chart") .ejChart({ series: [{ lineCap: 'butt' }]);}); </pre>	Not Applicable
Line join for series	<pre> Property:lineCap\$("#chart") .ejChart({ series: [{ lineJoin: 'round' }]);}); </pre>	Not Applicable
Opacity for series	<pre> Property:opacity\$("#chart") .ejChart({ series: [{ opacity: 0.7 }]);}); </pre>	<pre> Property:errorBarSett ingslet chart: Chart = new Chart({ series: [{ opacity: 0.7 }]);}); </pre>
Outlier settings of series	<pre> Property:outLierSettings\$("#ch art").ejChart({ series: [{ outLierSettings: { shape: 'rectangle' , size: { height: 30, width: 20} } }]);}); </pre>	Not Applicable
Palette	<pre> Property:palette\$("#chart") .ejChart({ series: [{ </pre>	<pre> Property:pointColor Mappinglet chart: Chart = </pre>

	<pre>palette: "ColorFieldName" }]);});</pre>	<pre>new Chart({ series: [{ pointColorMapping: 'color' }]);});chart.appendTo('#chart');</pre>
Positive fill for waterfall series	<pre>Property:positiveFill\$("#chart") .ejChart({ series: [{ positiveFill: "red" }]);});</pre>	<pre>Property:pointColorMappinglet chart: Chart = new Chart({ series: [{ pointColorMapping: 'color' }]);});chart.appendTo('#chart');</pre>
Show average value in box and whisker series	<pre>Property:showMedian\$("#chart") .ejChart({ series: [{ showMedian: true }]);});</pre>	<pre>Property:pointColorMappinglet chart: Chart = new Chart({ series: [{ showMean: false }]);});chart.appendTo('#chart');</pre>
To group the series of stacking collection.	<pre>Property:stackingGroup\$("#chart") .ejChart({ series: [{ stackingGroup: 'group' }]);});</pre>	<pre>Property:stackingGrouplet chart: Chart = new Chart({ series: [{ stackingGroup: 'group' }]);});chart.appendTo('#chart');</pre>
Specifies the type of the series to render in chart.	<pre>Property:type\$("#chart").ej Chart({ series: [{ type: 'Line' }]);});</pre>	<pre>Property:typelet chart: Chart = new Chart({ series: [{ type: 'Line' }]);});chart.appendTo('#chart');</pre>
Defines the visibility of the series.	<pre>Property:visibility\$("#chart") .ejChart({ series: [{ visibility: true }]);});</pre>	<pre>Property:visiblelet chart: Chart = new Chart({ series: [{ visible: true }]);});chart.appendTo('#chart');</pre>
Enables or disables	<pre>Property:visibleOnLegend \$("#chart").ejChart({ series: [{</pre>	<pre>Property:toggleVisibilitylet chart: Chart = new</pre>

		<p>the visibility of legend item.</p> <pre> visibleOnLegend : true }]);}); </pre> <p>Specifies the different types of spline curve.</p> <pre> Property:splineType \$("#chart").ejChart({ series: [{ splineType : 'Natural' }]);}); </pre> <p>Specifies the name of the x-axis that has to be associated with this series. Add an axis instance with this name to axes collection.</p> <pre> Property:xAxisName \$("#chart").ejChart({ series: [{ xAxisName : 'secondaryXAxis' }]);}); </pre> <p>Name of the property in the datasource that contains x value for the series.</p> <pre> Property:xName \$("#chart").ejChart({ series: [{ xName : 'x' }]);}); </pre> <p>Specifies the name of the y-</p> <pre> Property:yAxisName \$("#chart").ejChart({ series: [{ yAxisName : 'secondaryYAxis' }]);}); </pre> <p>Chart({ legendSettings: [{ toggleVisibility : true }]);});chart.appendTo('#chart');</p> <p>Property:splineType1</p> <pre> let chart: Chart = new Chart({ legendSettings: [{ splineType: 'Natural' }]);});chart.appendTo('#chart'); </pre> <p>Property:xAxisName1</p> <pre> let chart: Chart = new Chart({ series: [{ xAxisName: 'secondaryXAxis' }]);});chart.appendTo('#chart'); </pre> <p>Property:xName1</p> <pre> let chart: Chart = new Chart({ series: [{ xName: 'x' }]);});chart.appendTo('#chart'); </pre> <p>Property:yAxisName1</p> <pre> let chart: Chart = new Chart({ series: [{ </pre>
--	--	--

		<p>axis that has to be associated with this series.</p> <p>Add an axis instance with this name to axes collection.</p> <p>Name of the property in the datasource that contains y value for the series.</p> <p>Name of the property in the datasource that contains high value for the series.</p> <p>Name of the property in the datasource that contains low value for</p>	<pre> yAxisName: 'secondaryYAxis' });});chart.appendTo('#chart'); Property:yNamelet chart: Chart = new Chart({ series: [{ yName: 'y' }]);});chart.appendTo('#chart'); Property:highlet chart: Chart = new Chart({ series: [{ high: 'y' }]);});chart.appendTo('#chart'); Property:lowlet chart: Chart = new Chart({ series: [{ low: 'y' }]);});chart.appendTo('#chart'); </pre>
--	--	---	--

		<p>the series.</p> <p>Name of the property in the data source that contains close value for the series.</p> <p>Name of the property in the data source that contains open value for the series.</p> <p>Option to add trendline series to chart.</p> <p>Options for customizing the appearance of the series or data point while highlighting.</p>	<pre> Property:closelet chart: Chart = new Chart({ series: [{ close: 'y' }];});chart.appe ndTo('#chart'); Property:openlet chart: Chart = new Chart({ series: [{ open: 'y' }];});chart.appe ndTo('#chart'); Property:trendLines1 et chart: Chart = new Chart({ series: [{ trendLines : [{}] }];});chart.appe ndTo('#chart'); Property:highlightSettings\$("#c hart").ejChart({ series: [{ highlightSettings : {} }];}); </pre> <p>Not applicable.</p>
--	--	---	--

		<p>Options for customizing the appearance of the series/data point on selection.</p> <p>## marker</p> <p>visibility of marker</p> <p>Fill for marker</p> <p>Opacity for marker</p> <p>Shape of marker</p> <p>Property:selectionSettings <pre>\$("#chart").ejChart({ series: [{ selectionSettings : {} }];});</pre> </p> <p>Property:visible <pre>let chart: Chart = new Chart({ series: [{ marker: { visible: false } }];});chart.appendTo('#chart');</pre> </p> <p>Property:fill <pre>let chart: Chart = new Chart({ series: [{ marker: { fill : 'red' } }];});chart.appendTo('#chart');</pre> </p> <p>Property:opacity <pre>let chart: Chart = new Chart({ series: [{ marker: { opacity : 0.5 } }];});chart.appendTo('#chart');</pre> </p> <p>Property:shape <pre>let chart: Chart = new Chart({ series: [{ marker: { shape : 'Triangle' } }];});chart.appendTo('#chart');</pre> </p> <p>Not applicable.</p>
--	--	--

		<pre> Property:imageUrl let chart: Chart = new Chart({ series: [{ marker: { imageUrl : '' } }]); chart.appendTo('#chart'); </pre>
Image Url of marker	<pre> Property:imageUrl\$("#chart").ejChart({ series: [{ marker: { imageUrl : '' } }]); </pre>	<pre> Property:shape let chart: Chart = new Chart({ series: [{ marker: { border : { width: 2, color: 'red' } } }]); chart.appendTo('#chart'); </pre>
Border of marker	<pre> Property:border\$("#chart").ejChart({ series: [{ marker: { border : { width: 2, color: 'red' } } }]); </pre>	<pre> Property:height let chart: Chart = new Chart({ series: [{ marker: { height: 25 } }]); chart.appendTo('#chart'); </pre>
Height of marker	<pre> Property:size.height\$("#chart").ejChart({ series: [{ marker: { size: { height: 30 } } }]); </pre>	<pre> Property:width let chart: Chart = new Chart({ series: [{ marker: { width: 25 } }]); chart.appendTo('#chart'); </pre>
Width of marker	<pre> Property:size.width\$("#chart").ejChart({ series: [{ marker: { size: { width: 30 } } }]); </pre>	<pre> Property:width let chart: Chart = new Chart({ series: [{ marker: { width: 25 } }]); chart.appendTo('#chart'); </pre>
Width of marker	<pre> Property:size.width\$("#chart").ejChart({ series: [{ marker: { size: { width: 30 } } }]); </pre>	<pre> Property:marker.dataLabel let chart: Chart = new Chart({ series: [{ marker: { dataLabel: { } } }]); </pre>
DataLabel Setting of marker	<pre> Property:marker.dataLabel\$("#chart").ejChart({ series: [{ marker: { dataLabel: { } } }]); </pre>	<pre> Property:marker.dataLabel let chart: Chart = new Chart({ series: [{ marker: { dataLabel: { } } }]); </pre>

		<pre> });});chart.append endTo('#chart); Property:dataLabel. visiblelet chart: Chart = new Chart({ series: [{ marker: { dataLabel: { visible: true } } }]);});chart.append endTo('#chart); Property:dataLabel. namelet chart: Chart = new Chart({ series: [{ marker: { dataLabel: { name: ' ' } } }]);});chart.append endTo('#chart); Property:dataLabel. filllet chart: Chart = new Chart({ series: [{ marker: { dataLabel: { fill: 'pink' } } }]);});chart.append endTo('#chart); Property:dataLabel. filllet chart: Chart = new Chart({ series: [{ marker: { dataLabel: { opacity: 0.4 } } }]);});chart.append endTo('#chart); Property:dataLabel. positionlet chart: Chart = new Chart({ series: [{ marker: { dataLabel: { position: 'Top' } } </pre>
Visibility of dataLabel	<pre> Property:dataLabel.visible\$("#chart").ejChart({ series: [{ marker: {dataLabel: {visible: true } } }]);}); </pre>	<pre> });});chart.append endTo('#chart); Property:dataLabel. visiblelet chart: Chart = new Chart({ series: [{ marker: { dataLabel: { visible: true } } }]);});chart.append endTo('#chart); Property:dataLabel. namelet chart: Chart = new Chart({ series: [{ marker: { dataLabel: { name: ' ' } } }]);});chart.append endTo('#chart); Property:dataLabel. filllet chart: Chart = new Chart({ series: [{ marker: { dataLabel: { fill: 'pink' } } }]);});chart.append endTo('#chart); Property:dataLabel. filllet chart: Chart = new Chart({ series: [{ marker: { dataLabel: { opacity: 0.4 } } }]);});chart.append endTo('#chart); Property:dataLabel. positionlet chart: Chart = new Chart({ series: [{ marker: { dataLabel: { position: 'Top' } } </pre>
Text mapping name of dataLabel	<pre> Property:dataLabel.textMappingName\$("#chart").ejChart({ series: [{ marker: {dataLabel: {textMappingName: ' ' } } }]);}); </pre>	<pre> });});chart.append endTo('#chart); Property:dataLabel. namelet chart: Chart = new Chart({ series: [{ marker: { dataLabel: { name: ' ' } } }]);});chart.append endTo('#chart); Property:dataLabel. filllet chart: Chart = new Chart({ series: [{ marker: { dataLabel: { fill: 'pink' } } }]);});chart.append endTo('#chart); Property:dataLabel. filllet chart: Chart = new Chart({ series: [{ marker: { dataLabel: { opacity: 0.4 } } }]);});chart.append endTo('#chart); Property:dataLabel. positionlet chart: Chart = new Chart({ series: [{ marker: { dataLabel: { position: 'Top' } } </pre>
Fill color of data label	<pre> Property:dataLabel.fill\$("#chart").ejChart({ series: [{ marker: {dataLabel: {fill: 'pink' } } }]);}); </pre>	<pre> });});chart.append endTo('#chart); Property:dataLabel. namelet chart: Chart = new Chart({ series: [{ marker: { dataLabel: { name: ' ' } } }]);});chart.append endTo('#chart); Property:dataLabel. filllet chart: Chart = new Chart({ series: [{ marker: { dataLabel: { fill: 'pink' } } }]);});chart.append endTo('#chart); Property:dataLabel. filllet chart: Chart = new Chart({ series: [{ marker: { dataLabel: { opacity: 0.4 } } }]);});chart.append endTo('#chart); Property:dataLabel. positionlet chart: Chart = new Chart({ series: [{ marker: { dataLabel: { position: 'Top' } } </pre>
Opacity of data label	<pre> Property:dataLabel.opacity\$("#chart").ejChart({ series: [{ marker: {dataLabel: {opacity: 0.6 } } }]);}); </pre>	<pre> });});chart.append endTo('#chart); Property:dataLabel. namelet chart: Chart = new Chart({ series: [{ marker: { dataLabel: { name: ' ' } } }]);});chart.append endTo('#chart); Property:dataLabel. filllet chart: Chart = new Chart({ series: [{ marker: { dataLabel: { fill: 'pink' } } }]);});chart.append endTo('#chart); Property:dataLabel. filllet chart: Chart = new Chart({ series: [{ marker: { dataLabel: { opacity: 0.4 } } }]);});chart.append endTo('#chart); Property:dataLabel. positionlet chart: Chart = new Chart({ series: [{ marker: { dataLabel: { position: 'Top' } } </pre>
Text position of data label	<pre> Property:dataLabel.textPosition\$("#chart").ejChart({ series: [{ marker: {dataLabel: {textPosition: 'middle' } } }]);}); </pre>	<pre> });});chart.append endTo('#chart); Property:dataLabel. namelet chart: Chart = new Chart({ series: [{ marker: { dataLabel: { name: ' ' } } }]);});chart.append endTo('#chart); Property:dataLabel. filllet chart: Chart = new Chart({ series: [{ marker: { dataLabel: { fill: 'pink' } } }]);});chart.append endTo('#chart); Property:dataLabel. filllet chart: Chart = new Chart({ series: [{ marker: { dataLabel: { opacity: 0.4 } } }]);});chart.append endTo('#chart); Property:dataLabel. positionlet chart: Chart = new Chart({ series: [{ marker: { dataLabel: { position: 'Top' } } </pre>

		<pre> });});chart.appendTo('#chart'); </pre>
Alignment of data label	Property:dataLabel.verticalAlignment <pre> ent\$("#chart").ejChart({ series: [{ marker: {dataLabel: { verticalAlignment: 'near' } } }]);}); </pre>	Property:dataLabel.alignment <pre> let chart: Chart = new Chart({ series: [{ marker: { dataLabel: { alignment: 'Near' } } }]);});chart.appendTo('#chart'); </pre>
Border of data label	Property:dataLabel.border <pre> \$("#chart").ejChart({ series: [{ marker: {dataLabel: { border: { color: 'blue', width: 2, opacity: 0.4 } } } }]);}); </pre>	Property:dataLabel.alignment <pre> let chart: Chart = new Chart({ series: [{ marker: { dataLabel: { border: { color: 'blue', width: 2 } } } }]});});chart.appendTo('#chart'); </pre>
Offset for data label	Property:dataLabel.offset <pre> \$("#chart").ejChart({ series: [{ marker: {dataLabel: { offset: { x: 5, y: 6 } } } } }]);}); </pre>	Not Applicable
Margin of data label	Property:dataLabel.border <pre> \$("#chart").ejChart({ series: [{ marker: {dataLabel: { margin: { top: 10, bottom: 10, left: 10, right: 10} } } }]);}); </pre>	Property:dataLabel.margin <pre> let chart: Chart = new Chart({ series: [{ marker: { dataLabel: { margin: { top: 10, bottom: 10, left: 10, right: 10 } } } }]});});chart.appendTo('#chart'); </pre>
Font of data label	Property:dataLabel.border <pre> \$("#chart").ejChart({ series: [{ marker: {dataLabel: { font: { fontFamily: 'SegoeUI', fontStyle: 'italic', fontWeight: '600', opacity: 0.5, size: </pre>	Property:dataLabel.margin <pre> let chart: Chart = new Chart({ series: [{ marker: { dataLabel: { </pre>

		<pre> 12, color: 'red' }} } }};)); </pre>	<pre> font: { fontFamily: 'SegoeUI', fontStyle: 'italic', fontWeight: '600', opacity: 0.5, size: 12, color: 'red' }} } }};));chart.append endTo('#chart'); </pre>
	HTML templat e in dataLab el	<pre> Property:dataLabel.template\$("## chart").ejChart({ series: [{ marker: {dataLabel: { template: ' Chart ' } } }]]}); </pre>	<pre> Property:dataLabel. templatelet chart: Chart = new Chart({ series: [{ marker: {dataLabel: { template: ' Chart ' } } }]]});chart.append endTo('#chart'); </pre>
	Rounded corner radius X	Not Applicable	<pre> Property:dataLabel. rxlet chart: Chart = new Chart({ series: [{ marker: {dataLabel: { rx: 10 } } }]]});chart.append endTo('#chart'); </pre>
	Rounded corner radius Y	Not Applicable	<pre> Property:dataLabel. rylet chart: Chart = new Chart({ series: [{ marker: {dataLabel: { ry: 10 } } }]]});chart.append endTo('#chart'); </pre>
	Maximu m Label width for data label	<pre> Property:dataLabel.maximumLabel Width\$("##chart").ejChart({ series: [{ marker: {dataLabel: { maximumLabelWidth: 20}} } }]]}); </pre>	Not Applicable

		<p>Enable wrapping of text for data label Property: <code>dataLabel.enableWrap\$ ("#chart").ejChart({ series: [{ marker: {dataLabel: { enableWrap: true }} }]}]};</code> Not Applicable</p> <p>To show contrast color for data label Property: <code>dataLabel.showContrastColor\$ ("#chart").ejChart({ series: [{ marker: {dataLabel: { showContrastColor: true }} }]}]};</code> Not Applicable</p> <p>To show edge label for data label Property: <code>dataLabel.showEdgeLabels\$ ("#chart").ejChart({ series: [{ marker: {dataLabel: { showEdgeLabels: true }} }]}]};</code> Not Applicable</p> <p>## TrendLines</p> <p>Behaviour API in Essential JS 1 API in Essential JS 2</p> <p>Trend lines settings Property: <code>series.trendLines\$ ("#chart").ejChart({ series: [{ trendLines: []}]})</code>; Property: <code>series.trendLines</code> let chart: Chart = new Chart({ series: [{ trendLines: []}]}) ;chart.appendTo('#chart');</p> <p>Visibility of trend line Property: <code>trendLines.visibility\$ ("#chart").ejChart({ series: [{ trendLines: [visibility: true]}]})</code>; Not applicable</p> <p>Type of trend line Property: <code>trendLines.type\$ ("#chart").ejChart({ series: [{ trendLines: [type: 'linear']}]})</code>; Property: <code>trendLines.type</code> let chart: Chart = new Chart({ series: [{ trendLines: [type: 'Polynomial']}]}) ;chart.appendTo('#chart');</p> <p>Name of trend line Property: <code>trendLines.name\$ ("#chart").ejChart({ series: [{ trendLines: [name: 'trendLine']}]})</code>; Property: <code>trendLines.name</code> let chart: Chart = new Chart({ series: [{ trendLines: [name: 'trendLine']}]}) ;cha</p>
--	--	--

		<pre>rt.appendTo('#chart');</pre>
Period of trendLine	<pre>Property:trendLines.period\$ (" #chart").ejChart({ series: [{ trendLines: [period: 45]}]})</pre>	<pre>Property:trendLines.period let chart: Chart = new Chart({ series: [{ trendLines: [period: 45] }]});chart.append To('#chart');</pre>
Polynomial order for Polynomial trendLines	<pre>Property:trendLines.polynomialOrder\$ ("#chart").ejChart ({ series: [{ trendLines: [polynomialOrder: 3 }]})</pre>	<pre>Property:trendLines.polyno mialOrderlet chart: Chart = new Chart({ series: [{ trendLines: [polynomialOrder: 3] }]});chart.appendT o('#chart');</pre>
Backward forecast trendLines	<pre>Property:trendLines.backwardforecast\$ ("#chart").ejChar t({ series: [{ trendLines: [backwardforecast: 3 }]})</pre>	<pre>Property:trendLines.backw ardforecastlet chart: Chart = new Chart({ series: [{ trendLines: [backwardforecast: 3] }]});chart.appendT o('#chart');</pre>
Forward forecast trendLines	<pre>Property:trendLines.forwardForecast\$ ("#chart").ejChart ({ series: [{ trendLines: [forwardForecast: 3 }]})</pre>	<pre>Property:trendLines.forwar dForecastlet chart: Chart = new Chart({ series: [{ trendLines: [forwardForecast: 3] }]});chart.appendT o('#chart');</pre>
Fill trendLines	<pre>Property:trendLines.fill\$ ("#ch art").ejChart({ series: [{ trendLines: [fill: '#EEFFCC']}]})</pre>	<pre>Property:trendLines.filllet chart: Chart = new Chart({ series: [{ trendLines: [fill: 'EEFFCC']}]})</pre>
Width for	<pre>Property:trendLines.width\$ ("# chart").ejChart({</pre>	<pre>Property:trendLines.width let chart: Chart =</pre>

		<pre> trendLines: [{ trendLines: [width: 2]}]}}); </pre>	<pre> new Chart({ series: [{ trendLines: [width: 2]}]}});chart.appendTo('#chart'); </pre>
	<pre> Intercept value for trendLines </pre>	<pre> Property:trendLines.intercept\$let chart: Chart = new Chart({ series: [{ trendLines: [intercept: 2]}]}}); </pre>	<pre> Property:trendLines.intercept\$let chart: Chart = new Chart({ series: [{ trendLines: [intercept: 2]}]}});chart.appendTo('#chart'); </pre>
	<pre> Legend shape for trendLines </pre>	Not Applicable	<pre> Property:trendLines.legendShape\$let chart: Chart = new Chart({ series: [{ trendLines: [legendShape: 'Rectangle']}]}});chart.appendTo('#chart'); </pre>
	<pre> Animation settings for trendLines </pre>	Not Applicable	<pre> Property:trendLines.animation\$let chart: Chart = new Chart({ series: [{ trendLines: [animation: { enable: true }]}]}});chart.appendTo('#chart'); </pre>
	<pre> Marker settings for trendLines </pre>	Not Applicable	<pre> Property:trendLines.marker\$let chart: Chart = new Chart({ series: [{ trendLines: [{marker: { visible: true } }]}]}});chart.appendTo('#chart'); </pre>
	<pre> Tooltip for trendLines </pre>	<pre> Property:trendLines.tooltip\$let chart: Chart = new Chart({ series: [{ trendLines: [{ tooltip: { } }]}]}}); </pre>	<pre> Property:trendLines.enableTooltip\$let chart: Chart = new Chart({ series: [{ trendLines: [{enableTooltip: true }]}]}});chart.appendTo('#chart'); </pre>

		<p>Dash</p> <p>Array for trendLines</p> <p>Property:trendLines.dashArray</p> <pre>\$("#chart").ejChart({ series: [{ trendLines: [{ dashArray: '10, 5' }] }]});</pre> <p>Not Applicable.</p> <p>Visible on legend</p> <p>Property:trendLines.visibleOnLegend</p> <pre>\$("#chart").ejChart({ series: [{ trendLines: [{ visibleOnLegend: true }] }]});</pre> <p>Not Applicable.</p> <p>## Striplines</p> <p>Behaviour</p> <p>API in Essential JS 1</p> <p>API in Essential JS 2</p> <p>Default behaviour for striplines</p> <p>Property:primaryXAxis.striplines</p> <pre>let chart: Chart = new Chart({ primaryXAxis: { striplines: [{ visible: true }] } }); chart.appendTo('#chart');</pre> <p>border for stripline</p> <p>Property:striplines.borderColor</p> <pre>let chart: Chart = new Chart({ primaryXAxis: { striplines: [{ border: { color: 'red', width: 2 } }] } }); chart.appendTo('#chart');</pre> <p>Background color for stripline</p> <p>Property:striplines.borderColor</p> <pre>let chart: Chart = new Chart({ primaryXAxis: { striplines: [{ color: 'red' }] } }); chart.appendTo('#chart');</pre> <p>Start value for</p> <p>Property:striplines.start</p> <pre>let chart: Chart = new Chart({ primaryXAxis: {</pre>
--	--	--

		<pre> stripline stripLines: [{ start: stripLines: [{ 10 }]]]); start: 5}}]);chart.appendTo('#chart'); </pre>
End value for stripline	<pre> Property:stripLines.end\$("#chart").ejChart({ primaryXAxis: { stripLines: [{ end: 10 }]]]); </pre>	<pre> Property:stripLines.endlet chart: Chart = new Chart({ primaryXAxis: { stripLines: [{ end: 5}}]);chart.appendTo('#chart'); </pre>
Start from Axis for stripline	<pre> Property:stripLines.startFrom Axis\$("#chart").ejChart ({ primaryXAxis: { stripLines: [{ startFromAxis: true }}]); </pre>	<pre> Property:stripLines.startFromAxislet chart: Chart = new Chart({ primaryXAxis: { stripLines: [{ startFromAxis: true}}]);chart.appendTo('#chart'); </pre>
Text in stripline	<pre> Property:stripLines.text\$("#chart").ejChart({ primaryXAxis: { stripLines: [{ text: 'StripLine; }]]]); </pre>	<pre> Property:stripLines.textlet chart: Chart = new Chart({ primaryXAxis: { stripLines: [{ text: 'stripline'}}]);chart.appendTo('#chart'); </pre>
Text alignment in stripline	<pre> Property:stripLines.textAlignment\$("#chart").ejChart({ primaryXAxis: { stripLines: [{ textAlignment: 'Far; }}]); </pre>	<pre> Property:stripLines.horizontalAlignmentlet chart: Chart = new Chart({ primaryXAxis: { stripLines: [{ horizontalAlignment: 'Far'}}]);chart.appendTo('#chart'); </pre>
Vertical Text alignment in stripline	Not Applicable	<pre> Property:stripLines.verticalAlignmentlet chart: Chart = new Chart({ primaryXAxis: { stripLines: [{ verticalAlignment: 'Far'}}]);chart.appendTo('#chart'); </pre>
Size of stripline	<pre> Property:stripLines.width\$("#chart").ejChart({ primaryXAxis: { stripLines: [{ width: 10; }]]]); </pre>	<pre> Property:stripLines.sizelet chart: Chart = new Chart({ primaryXAxis: { stripLines: [{ </pre>

		<pre> size: 10 }}}});chart.appendTo ('#chart'); Property:stripLines.sizelet chart: Chart = new Chart({ primaryXAxis: { stripLines: [{ zIndex: 'Behind' }}}});chart.appendTo ('#chart'); Property:stripLines.textStyl elet chart: Chart = new Chart({ primaryXAxis: { stripLines: [{ textStyle: { }}}});chart.appendTo ('#chart'); </pre>
ZIndex of stripli ne	<pre> Property:stripLines.zIndex\$(" #chart").ejChart({ primaryXAxis: { stripLines: [{ zIndex: 'Behind' }}}}); </pre>	
Font style of stripli ne	<pre> Property:stripLines.fontStyle\$ ("#chart").ejChart({ primaryXAxis: { stripLines: [{ fontStyle: { }]}}}); </pre>	
## Multilevel Labels		
Behavi our	API in Essential JS 1	API in Essential JS 2
Defaul t behavi our for multil evellLa bels	<pre> Property:primaryXAxis.multilev elLabels\$("#chart").ejCha rt({ primaryXAxis: { multilevelLabels: [{ visible: true }]}}}); </pre>	<pre> Property:primaryXAxis.m ultilevelLabelslet chart: Chart = new Chart({ primaryXAxis: { multilevelLabels: [{ }}}});chart.appendT o('#chart'); </pre>
Defaul t behavi our for multil evellLa bels	<pre> Property:primaryXAxis.multilev elLabels\$("#chart").ejCha rt({ primaryXAxis: { multilevelLabels: [{ visible: true }]}}}); </pre>	<pre> Property:primaryXAxis.m ultilevelLabelslet chart: Chart = new Chart({ primaryXAxis: { multilevelLabels: [{ }}}});chart.appendT o('#chart'); </pre>
Text alignm ent for multil evellLa bels	<pre> Property:multiLevelLabels.text Alignment\$("#chart").ejC hart({ primaryXAxis: { multilevelLabels: [{ textAlignment: 'Near' }}}}); </pre>	<pre> Property:multilevelLabels .alignmentlet chart: Chart = new Chart({ primaryXAxis: { multilevelLabels: [{alignment: 'Near' </pre>

		<pre>]]]]);chart.appendT o('#chart'); </pre>
Text overfl ow for multil evelLa bels	Property:multiLevelLabels.text Overflow\$("#chart").ejCh art({ primaryXAxis: { multilevelLabels: [{ textOverFlow: 'Trim' }]}}]);	Property:multiLevelLabels .overflow let chart: Chart = new Chart({ primaryXAxis: { multilevelLabels: [{overflow: 'Trim' }]}});chart.appendT o('#chart');
Borde r for multil evelLa bels	Property:multiLevelLabels.bord er\$("#chart").ejChart({ primaryXAxis: { multilevelLabels: [{ border: { width: 2, color: 'red', type: 'brace' }]}}]);	Property:multiLevelLabels .border let chart: Chart = new Chart({ primaryXAxis: { multilevelLabels: [{ border: { width: 2, color: 'red', type: 'brace' } }]}]);chart.appendT o('#chart');
Start value for label	Property:multiLevelLabels.start \$("#chart").ejChart({ primaryXAxis: { multilevelLabels: [{ start: 45 }]}}]);	Property:multiLevelLabels .categories.start let chart: Chart = new Chart({ primaryXAxis: { multilevelLabels: [{ categories: [{ start: 45}] }]}]);chart.appendT o('#chart');
End value for label	Property:multiLevelLabels.start \$("#chart").ejChart({ primaryXAxis: { multilevelLabels: [{ end: 45 }]}}]);	Property:multiLevelLabels .categories.end let chart: Chart = new Chart({ primaryXAxis: { multilevelLabels: [{ categories: [{ end: 45}] }]}]);chart.appendT o('#chart');
Text for label	Property:multiLevelLabels.text \$("#chart").ejChart({ primaryXAxis: { multilevelLabels: [{ text: 'Start' }]}}]);	Property:multiLevelLabels .categories.text let chart: Chart = new Chart({ primaryXAxis: { multilevelLabels: [{ categories: [{ text: 'text' }] }]}]);chart.appendT o('#chart');

		<p>Property:multiLevelLabels.categories.maximumTextWidth</p> <p>let chart: Chart = new Chart({ primaryXAxis: { multilevelLabels: [{ categories: [{ maximumTextWidth: 20 }] }] } }); chart.appendTo('#chart');</p> <p>Property:multiLevelLabels.level</p> <p>\$("#chart").ejChart({ primaryXAxis: { multilevelLabels: [{ level: 2 }] } });</p> <p>## Methods</p> <p>API in Essential JS 1 API in Essential JS 2</p> <p>Property:chart.animate</p> <p>\$("#chart").ejChart({ animate: () { } });</p> <p>Property:chart.refresh()</p> <p>let chart: Chart = new Chart({}); chart.appendTo('#chart'); chart.width = '400'; chart.refresh();</p> <p>Property:chart.export()</p> <p>let chart: Chart = new Chart({}); chart.export('J PEG', 'chart'); chart.appendTo('#chart');</p> <p>Property:chart.print()</p> <p>let chart: Chart = new Chart({}); chart.print('chart'); chart.appendTo('#chart');</p> <p>Property:chart.addSeries()</p> <p>let chart: Chart = new Chart({}); chart.appendTo('#chart'); chart.addSeries();</p>
maximum text width for label	Property:multiLevelLabels.maximumTextWidth	Not applicable.
level of labels	Property:multiLevelLabels.level	Categories are used
Behaviour	API in Essential JS 1	API in Essential JS 2
animation for series	Property:chart.animate	Not applicable
Redraw for chart	Property:chart.redraw	Property:chart.refresh()
Export	Property:chart.export()	Property:chart.export()
Print	Property:chart.print()	Property:chart.print()
AddSeries	Not Applicable	Property:chart.addSeries()

		<p>Removes series from the chart. Not applicable in Essential JS 1.</p> <p>Property: <code>chart.removeSeries()</code></p> <pre>let chart: Chart = new Chart({}); chart.appendTo('#chart'); chart.removeSeries();</pre> <p>## Events</p> <p>Behavior</p> <p>API in Essential JS 1</p> <p>API in Essential JS 2</p> <p>Fires on annotation click</p> <p>Property: <code>annotationClick\$</code></p> <pre>("#chart").ejChart({ annotationClick: () { } });</pre> <p>Not applicable</p> <p>Fires after animation</p> <p>Property: <code>animationComplete\$</code></p> <pre>("#chart").ejChart({ animationComplete: () { } });</pre> <p>Property: <code>animationComplete()</code></p> <pre>let chart: Chart = new Chart({ animationComplete: () => { } }); chart.appendTo('#chart');</pre> <p>Fires on axis label click</p> <p>Property: <code>axisLabelClick\$</code></p> <pre>("#chart").ejChart({ axisLabelClick: () { } });</pre> <p>Not applicable</p> <p>Fires before axis label render</p> <p>Property: <code>axisLabelRendering\$</code></p> <pre>("#chart").ejChart({ axisLabelRendering: () { } });</pre> <p>Property: <code>axisLabelRender()</code></p> <pre>let chart: Chart = new Chart({ axisLabelRender: () => { } }); chart.appendTo('#chart');</pre> <p>Fires on axis label mouse move</p> <p>Property: <code>axisLabelMouseMove\$</code></p> <pre>("#chart").ejChart({ axisLabelMouseMove: () { } });</pre> <p>Not applicable</p> <p>Fires on axis label initialize</p> <p>Property: <code>axisLabelInitialize\$</code></p> <pre>("#chart").ejChart({ axisLabelInitialize: () { } });</pre> <p>Not applicable</p>
--	--	---

		<p>Fires before Property:axesRangeCalculate\$("#chart").ejChart({ axis axesRangeCalculate: () { range } calcula } tion } });</p> <p>Fires on axis Property:axisTitleRendering\$("#chart").ejChart({ title axisTitleRendering: () { render } ing } });</p> <p>Fires on Property:afterResize\$("#chart").ejChart({ after afterResize: () { chart } resize } });</p> <p>Fires on Property:beforeResize\$("#chart").ejChart({ before beforeResize: () { chart } resize } });</p> <p>Fires on Property:chartClick\$("#chart").ejChart({ chart chartClick: () { click } });</p> <p>Fires on Property:chartMouseMove\$("#chart").ejChart({ chart chartMouseMove: () { mouse } move } });</p> <p>Fires on Property:chartMouseLeave\$("#chart").ejChart({ chart chartMouseLeave: () { mouse } leave } });</p>	<p>Property:axisRangeCalculated() let chart: Chart = new Chart({ axisRangeCalculated: () => { });chart.appendTo('#chart');</p> <p>Not applicable</p> <p>Not applicable</p> <p>Property:resized let chart: Chart = new Chart({ resized: () => { });chart.appendTo('#chart');</p> <p>Property:chartMouseClick let chart: Chart = new Chart({ chartMouseClick: () => { });chart.appendTo('#chart');</p> <p>Property:chartMouseMove let chart: Chart = new Chart({ chartMouseMove: () => { });chart.appendTo('#chart');</p> <p>Property:chartMouseLeave let chart: Chart = new Chart({ chartMouseLeave: () => { }</p>
--	--	---	--

		<pre> });chart.append To('#chart'); Fires on Property:chartDoubleClick\$("#cha before rt").ejChart({ chart chartDoubleClick: () { double }}); click </pre>	<pre> });chart.append To('#chart'); Fires on Property:chartDoubleClick\$("#cha before rt").ejChart({ chart chartDoubleClick: () { double }}); click </pre>	Not applicable
		<pre> Fires on Property:chartmouse chart Not Applicable mouse up </pre>	<pre> Property:chartmouse Up let chart: Chart = new Chart({ chartmouseUp: () => { }});chart.append To('#chart'); </pre>	
		<pre> Fires on Property:chartmouse chart Not Applicable mouse down </pre>	<pre> Property:chartmouse Down let chart: Chart = new Chart({ chartmouseDown: () => { }});chart.append To('#chart'); </pre>	
		<pre> Fires during the calcula tion of chart area bound s. You can use this event to custo mize the bound s of chart area </pre>	<pre> Property:chartAreaBoundsCalculat e\$("#chart").ejChart({ chartAreaBoundsCalculate: () { }}); </pre>	Not applicable

		<p>Fires when the dragging is started</p> <pre>Property:dragStart\$("#chart").ejChart({ dragStart: () { Not applicable }});</pre> <p>Fires while dragging</p> <pre>Property:dragging\$("#chart").ejChart({ dragging: () { Not applicable }});</pre> <p>Fires when the dragging is completed</p> <pre>Property:dragCompletelet chart: Chart = new Chart({ dragComplete: () => { }});chart.append To('#chart');</pre> <p>Fires when chart is destroyed.</p> <pre>Property:destroy\$("#chart").ejChart({ destroy: () { Not applicable }});</pre> <p>Fires after chart is created.</p> <pre>Property:loadedlet chart: Chart = new Chart({ loaded: () => { }});chart.append To('#chart');</pre> <p>Fires before rendering the data labels.</p> <pre>Property:textRenderlet chart: Chart = new Chart({ textRender: () => { }});chart.append To('#chart');</pre> <p>Fires when error bar is</p> <pre>Property:errorBarRendering\$("#c hart").ejChart({ errorBarRendering: () { Not applicable }});</pre>
--	--	--

		<p>rendering.</p> <p>Fires during the calculation of legend bounds.</p> <p>Property:legendBoundsCalculate\$ ("#chart").ejChart({ legendBoundsCalculate: () { }});</p> <p>Not applicable</p> <p>Fires on clicking the legend item.</p> <p>Property:legendItemClick\$ ("#chart").ejChart({ legendItemClick: () { }});</p> <p>Not applicable</p> <p>Fires when moving mouse over legend item.</p> <p>Property:legendItemMouseMove\$ ("#chart").ejChart({ legendItemMouseMove: () { }});</p> <p>Not applicable</p> <p>Fires before rendering the legend item.</p> <p>Property:legendRenderlet chart: let chart: Chart = new Chart({ legendRender: () => { }});chart.appendTo('#chart');</p> <p>Fires before loading the chart.</p> <p>Property:load\$ ("#chart").ejChart({ load: () { }});</p> <p>Property:loadlet chart: Chart = new Chart({ load: () => { }});chart.appendTo('#chart');</p> <p>Fires, when multi level labels are</p> <p>Property:multiLevelLabelRendering\$ ("#chart").ejChart({ multiLevelLabelRendering: () { }});</p> <p>Property:axisMultiLabelRenderlet chart: Chart = new Chart({ axisMultiLabelRender : () => {</p>
--	--	---

		<pre> rendering. Property:pointClick let chart: Chart = new Chart({ pointClick : () => { });chart.append To('#chart'); Fires on clicking a point in chart. Property:pointRegionClick\$("#chart").ejChart({ pointRegionClick: () { }); Fires when mouse is moved over a point. Property:pointRegionMouseMove\$("#chart").ejChart({ pointRegionMouseMove: () { }); Fires before rendering chart. Property:preRender\$("#chart").ejChart({ preRender: () { }); Fires when point render . Property:pointRender let chart: Chart = new Chart({ pointRender : () => { });chart.append To('#chart'); Fires after selected the data in chart. Property:rangeSelected\$("#chart").ejChart({ rangeSelected: () { }); Fires after selecting a series. Property:seriesRegionClick\$("#chart").ejChart({ seriesRegionClick: () { }); Fires before render Property:seriesRendering\$("#chart").ejChart({ seriesRendering: () { }); </pre>	<pre> });chart.append To('#chart'); Property:pointClick let chart: Chart = new Chart({ pointClick : () => { });chart.append To('#chart'); Property:pointMove let chart: Chart = new Chart({ pointMove : () => { });chart.append To('#chart'); Not applicable Property:pointRender let chart: Chart = new Chart({ pointRender : () => { });chart.append To('#chart'); Not applicable Not applicable Property:seriesRende let chart: Chart = new Chart({ </pre>
--	--	--	---

		<pre> ing a series. Fires before render ing the marke r symbo ls. Fires before render ing the trendli ne Fires before render ing the Chart title. Fires before render ing the Chart sub title. Fires before render ing the tooltip . Fires before render ing crossh air </pre>	<pre> seriesRender : () => { }});chart.append To('#chart'); Property:symbolRendering\$("#ch art").ejChart({ symbolRendering: () { }}); Property:trendlineRendering\$("#c hart").ejChart({ trendlineRendering: () { Property:titleRendering\$("#chart ").ejChart({ titleRendering: () { }}); Property:subTitleRendering\$("#ch art").ejChart({ subTitleRendering: () { Property:tooltipRender er let chart: Chart = new Chart({ tooltipRender : () => { }});chart.append To('#chart'); Property:trackAxisToolTip\$("#cha rt").ejChart({ trackAxisToolTip: () { }}); </pre>	<p>Not applicable</p> <p>Not applicable</p> <p>Not applicable</p> <p>Not applicable</p> <p>Not applicable</p> <p>Property:tooltipRender</p> <p>Not applicable</p>
--	--	--	---	---

		<p>tooltip in axis</p> <p>Fires before render ing trackb all tooltip .</p> <p>Event trigger ed when scroll starts.</p> <p>Event trigger ed when scroll ends.</p> <p>Event trigger ed when scroll chang es.</p> <p>Fires while perfor ming rectan gle zoomi ng in chart.</p> <p>Behaviour API in Essential JS 1</p> <p>selected data index</p>	<p>Property:trackToolTip\$("#chart").ejChart({ trackToolTip: Not applicable () { } });</p> <p>Property:scrollStart let chart: Chart = new Chart({ scrollStart : (=> { } });chart.append To('#chart');</p> <p>Property:scrollEndle t chart: Chart = new Chart({ scrollEnd: (=> { } });chart.append To('#chart');</p> <p>Property:scrollChang elet chart: Chart = new Chart({ scrollChange: (=> { } });chart.append To('#chart');</p> <p>Property:zoomCompl etelet chart: Chart = new Chart({ zoomComplete: (=> { } });chart.append To('#chart');</p> <p>## Chart properties</p> <p>Property:selectedDataPo intIndexes\$("#chart").ejChart({</p> <p>Property:selectedDataIndex eslet chart: Chart = new</p>
--	--	---	---

		<pre>selectedDataPointIndexes: [{ seriesIndex: 0, pointIndex: 1}]); Chart({selectedDataIndexes: [{ series: 0, point: 1}]);chart.appendTo('#chart');</pre>
sideBySideSeriesPlacement for column based series	<pre>Property:sideBySideSeriesPlacement:\$("#chart").ejChart({ sideBySideSeriesPlacement});</pre>	<pre>Property:sideBySidePlacement:let chart: Chart = new Chart({ sideBySidePlacement: true});chart.appendTo('#chart');</pre>
ZoomSettings	<pre>Property:zooming:\$("#chart").ejChart({ zooming: { enable: true, enableDeferredZooming: true, enablePinch: true, enableMouseWheel: true, enableScrollBar: true, toolbarItems: [], type: 'X' });</pre>	<pre>Property:zoomSettings:let chart: Chart = new Chart({ zoomSettings: { enable: true, enablePinchZooming: true, enableDeferredZooming: true enableMouseWheelZooming: true, enableSelectionZooming: true, enableScrollBar: true });chart.appendTo('#chart');</pre>
Background color of the chart	<pre>Property:background\$("#container").ejChart({ background: 'transparent'});</pre>	<pre>Property:background:let chart: Chart = new Chart({ background: '#EEFFCC'});chart.appendTo('#chart');</pre>
URL of the image to be used as chart background.	<pre>Property:backGroundImageUrl\$("#container").ejChart({ backGroundImageUrl: '../images/chart/weight.png'});</pre>	Not Applicable
Customizing border of the chart	<pre>Property:border\$("#container").ejChart({ border: { width: 2, color: '#CCEEFF', opacity: 0.5}});</pre>	<pre>Property:border:let chart: Chart = new Chart({ border: { width: 2, color: '#CCEEFF'}});chart.appendTo('#chart');</pre>
This provides	<pre>Property:exportSettings\$("#container").ejChart({</pre>	<pre>Property:export():let chart: Chart = new Chart({ border: {</pre>

		<div> <div>options for customizing export settings</div> <div> <pre>exportSettings: { filename : "chart", angle: '45' }}; Property:chartArea\$("#container").ejChart({ chartArea: { background: 'transparent', border: { opacity: 0.3, color: 'red', width: 2 } } });</pre> </div> </div> <div> <div>Property:chartArea</div> <div> <pre>let chart: Chart = new Chart({ chartArea: { background: 'transparent', border: { width: 2, color: '#CCEEFF' } } }); chart.appendTo('#chart'); chart.export(type, fileName);</pre> </div> </div>
--	--	---

<tr>

<td>Behaviour</td>

<td>API in Essential JS 1</td>

<td>API in Essential JS 2</td>

</tr>

<tr>

<td>Alternate grid band</td>

<td>

Property:<i>alternateGridBand </i>

</br>

</br>

<code>

```
$("#container").ejChart({
  axes: [ { alternateGridBand: { even: { fill: 'red' }}}]
});
```

</code>

</td>

<td>

Not applicable

</td>

</tr>

```

<tr>
<td><b>Axis line cross value</b></td>
<td>
<b>Property</b><i>crossesAt</i>
</br>
</br>
<code>
$( "#container" ).ejChart({
axes: [ { crossesAt: 0 } ]
});
</code>
</td>
<td>
<b>Property</b><i>crossesAt</i>
</br>
</br>
<code>
let chart: Chart = new Chart({
axes: [ { crossesAt: 4 } ]
});
chart.appendTo('#chart');
</code>
</td>
</tr>
<tr>
<td><b>axis name with which the axis line has to be crossed</b></td>
<td>
<b>Property</b><i>crossesInAxis</i>
</br>
</br>
<code>
$( "#container" ).ejChart({
axes: [ { crossesInAxis: " " } ]

```

```

});
</code>
</td>
<td>
<b>Property</b>:<i>crossesInAxis</i>
</br>
</br>
<code>
let chart: Chart = new Chart({
axes: [ { crossesInAxis: " } ]
});
chart.appendTo('#chart');
</code>
</td>
</tr>
<tr>
<td><b>axis elements placed with axis line</b></td>
<td>
<b>Property</b>:<i>showNextToAxisLine </i>
</br>
</br>
<code>
$("#container").ejChart({
axes: [ { showNextToAxisLine : true } ]
});
</code>
</td>
<td>
<b>Property</b>:<i>placeNextToAxisLine</i>
</br>
</br>
<code>
let chart: Chart = new Chart({

```

```

axes: [ { placeNextToAxisLine: " }
});
chart.appendTo('#chart');
</code>
</td>
</tr>
<tr>
<td><b>axis line style</b></td>
<td>
<b>Property</b>:<i>axisLine.color</i>
</br>
</br>
<code>
$("#container").ejChart({
axes: [ { axisLine: { color : 'red' } } ]
});
</code>
</td>
<td>
<b>Property</b>:<i>lineStyle.color</i>
</br>
</br>
<code>
let chart: Chart = new Chart({
axes: [ { lineStyle: { color: 'black' } } ]
});
chart.appendTo('#chart');
</code>
</td>
</tr>
<tr>
<td><b>axis line dashArray</b></td>
<td>

```



```

<b>Property</b>:<i>axisLine.color</i>
</br>
</br>
<code>
$("#container").ejChart({
axes: [ { axisLine: { dashArray : '10, 5' } }]
});
</code>
</td>
<td>
<b>Property</b>:<i>lineStyle.dashArray</i>
</br>
</br>
<code>
let chart: Chart = new Chart({
axes: [ { lineStyle: { dashArray: '10, 5' } }]
});
chart.appendTo('#chart');
</code>
</td>
</tr>
<tr>
<td><b>Offset for axis</b></td>
<td>
<b>Property</b>:<i>axisLine.offset</i>
</br>
</br>
<code>
$("#container").ejChart({
axes: [ { axisLine: { offset : 10 } }]
});
</code>
</td>

```

<pre><td> Property:<i>plotOffset</i> </br> </br> <code> let chart: Chart = new Chart({ axes: [{ plotOffset: 10 }] }); chart.appendTo('#chart'); </code> </td> </tr> <tr> <td>Visible of an axis</td> <td> Property:<i>axisLine.offset</i> </br> </br> <code> \$("#container").ejChart({ axes: [{ axisLine: { visible : false } }] }); </code> </td> <td> Property:<i>visible</i> </br> </br> <code> let chart: Chart = new Chart({ axes: [{ visible: false }] }); chart.appendTo('#chart');</pre>

```
</code>
</td>
</tr>
<tr>
<td><b>Width of an axis</b></td>
<td>
<b>Property</b>:<i>axisLine.width</i>
<br>
<br>
<code>
$( "#container" ).ejChart({
axes: [ { axisLine: { width : 2 } } ]
});
</code>
</td>
<td>
<b>Property</b>:<i>lineStyle.width</i>
<br>
<br>
<code>
let chart: Chart = new Chart({
axes: [ { lineStyle: { width: 3 } } ]
});
chart.appendTo('#chart');
</code>
</td>
</tr>
<tr>
<td><b>Column index of an axis</b></td>
<td>
<b>Property</b>:<i>columnIndex</i>
<br>
<br>
</td>
```

```
<code>
```

```
$("#container").ejChart({
  axes: [ { columnIndex: 2 } ]
});
```

```
</code>
```

```
</td>
```

```
<td>
```

```
<b>Property</b>:<i>columnIndex</i>
```

```
</br>
```

```
</br>
```

```
<code>
```

```
let chart: Chart = new Chart({
  axes: [ { columnIndex: 2 } ]
});
chart.appendTo('#chart');
```

```
</code>
```

```
</td>
```

```
</tr>
```

```
<tr>
```

```
<td><b>span of an axis to place horizontally or vertically</b></td>
```

```
<td>
```

```
<b>Property</b>:<i>columnSpan</i>
```

```
</br>
```

```
</br>
```

```
<code>
```

```
$("#container").ejChart({
  axes: [ { columnIndex: 2 } ]
});
```

```
</code>
```

```
</td>
```

```
<td>
```

```
<b>Property</b>:<i>span</i>
```

```
</br>
```

</br>

<code>

let chart: Chart = new Chart({

axes: [{ span: 2 }]

});

chart.appendTo('#chart');

</code>

</td>

</tr>

<tr>

<td>Crosshair label of an axis</td>

<td>

Property:<i>crossHairLabel.visible</i>

</br>

</br>

<code>

\$("#container").ejChart({

axes: [{ crossHairLabel: { visible: true } }]

});

</code>

</td>

<td>

Property:<i>crossHairTooltip.enable</i>

</br>

</br>

<code>

let chart: Chart = new Chart({

axes: [{ crossHairTooltip: { enable: true } }]

});

chart.appendTo('#chart');

</code>

</td>

</tr>

<tr> <td>Crosshair label color of an axis</td>
--

<td> Not applicable

</td> <td> Property<i>crossHairTooltip.fill</i>
--

</br> </br> <code> let chart: Chart = new Chart({ axes: [{ crossHairTooltip: { fill: 'red' } }] }); chart.appendTo('#chart');
--

</code>

</td>

</tr>

<tr> <td>Crosshair label text style</td>
--

<td> Not applicable

</td>

<td> Property<i>crossHairTooltip.textStyle</i>
--

</br>

</br>

<code> let chart: Chart = new Chart({ axes: [{ crossHairTooltip: { textStyle: { } } }] }); chart.appendTo('#chart');

</code>

</td>

```
</tr>
<tr>
<td><b>Desired interval count for primaryYAxis</b></td>
<td>
<b>Property</b>:<i>desiredIntervals</i>
</br>
</br>
<code>
$("#container").ejChart({
axes: [ { desiredIntervals: 4}]
});
</code>
</td>
<td>
<b>Property</b>:<i>desiredIntervals</i>
</br>
</br>
<code>
let chart: Chart = new Chart({
axes: [ { desiredIntervals: 4 }]
});
chart.appendTo('#chart');
</code>
</td>
</tr>
<tr>
<td><b>Edges primaryYAxis</b></td>
<td>
<b>Property</b>:<i>edgeLabelPlacement</i>
</br>
</br>
<code>
$("#container").ejChart({
```

```
axes: [ { edgeLabelPlacement: 'none' } ]
});
</code>
</td>
<td>
<b>Property</b>:<i>edgeLabelPlacement</i>
</br>
</br>
<code>
let chart: Chart = new Chart({
axes: [ { edgeLabelPlacement: 'Shift' } ]
});
chart.appendTo('#chart');
</code>
</td>
</tr>
<tr>
<td><b>Enables trim for axis labels</b></td>
<td>
<b>Property</b>:<i>enableTrim</i>
</br>
</br>
<code>
$("#container").ejChart({
axes: [ { enableTrim: true } ]
});
</code>
</td>
<td>
<b>Property</b>:<i>enableTrim</i>
</br>
</br>
<code>
```



```
let chart: Chart = new Chart({
  axes: [ { enableTrim: true } ]
});
chart.appendTo('#chart');
```

</code>

</td>

</tr>

<tr>

<td>Specifies the interval of the axis according to the zoomed data of the chart</td>

<td>

Property:<i>enableAutoIntervalOnZooming</i>

</br>

</br>

<code>

```
$("#container").ejChart({
  axes: [ { enableAutoIntervalOnZooming: true } ]
});
```

</code>

</td>

<td>

Property:<i>enableAutoIntervalOnZooming</i>

</br>

</br>

<code>

```
let chart: Chart = new Chart({
  axes: [ { enableAutoIntervalOnZooming: true } ]
});
```

```
chart.appendTo('#chart');
```

</code>

</td>

</tr>

<tr>

<td>Specifies the interval of the axis according to the zoomed data of the chart</td>

```

<td>
<b>Property</b>:<i>enableAutoIntervalOnZooming</i>
</br>
</br>
<code>
$("#container").ejChart({
axes: [ { enableAutoIntervalOnZooming: true } ]
});
</code>
</td>

```

```

<td>
<b>Property</b>:<i>enableAutoIntervalOnZooming</i>
</br>
</br>
<code>
let chart: Chart = new Chart({
axes: [ { enableAutoIntervalOnZooming: true } ]
});
chart.appendTo('#chart');
</code>
</td>
</tr>

```

```

<td><b>Font style for primaryYAxis</b></td>

```

```

<td>
<b>Property</b>:<i>font</i>
</br>
</br>
<code>
$("#container").ejChart({
axes: [ { font: { fontFamily: 'Calibri', fontStyle: 'italic', fontWeight: '', opacity: 0.5, size: 12 } } ]
});
</code>

```

```
</td>
<td>
<b>Property</b>:<i>titleStyle</i>
</br>
</br>
<code>
let chart: Chart = new Chart({
axes: [ { titleStyle: { } } ]
});
chart.appendTo('#chart');
</code>
</td>
</tr>
<tr>
<td><b>Indexed for category axis</b></td>
<td>
<b>Property</b>:<i>isIndexed</i>
</br>
</br>
<code>
$("#container").ejChart({
axes: [ { isIndexed: true } ]
});
</code>
</td>
<td>
<b>Property</b>:<i>isIndexed</i>
</br>
</br>
<code>
let chart: Chart = new Chart({
axes: [ { isIndexed: true } ]
});
```

```

chart.appendTo('#chart');
</code>
</td>
</tr>
<tr>
<td><b>Interval type for date time axis</b></td>
<td>
<b>Property</b><i>intervalType</i>
</br>
</br>
<code>
$( "#container" ).ejChart({
axes: [ { intervalType: 'Auto' } ]
});
</code>
</td>
<td>
<b>Property</b><i>intervalType</i>
</br>
</br>
<code>
let chart: Chart = new Chart({
axes: [ { intervalType: 'Auto' } ]
});
chart.appendTo('#chart');
</code>
</td>
</tr>
<tr>
<td><b>Inversed axis</b></td>
<td>
<b>Property</b><i>isInversed</i>
</br>

```

</br>

<code>

```
$("#container").ejChart({
  axes: [ { isInversed: true } ]
});
```

</code>

</td>

<td>

Property:<i>isInversed</i>

</br>

</br>

<code>

```
let chart: Chart = new Chart({
  axes: [ { isInversed: true } ]
});
```

```
chart.appendTo('#chart');
```

</code>

</td>

</tr>

<tr>

<td>Custom label format</td>

<td>

Property:<i>labelFormat</i>

</br>

</br>

<code>

```
$("#container").ejChart({
  axes: [ { labelFormat: '{value}K' } ]
});
```

</code>

</td>

<td>

Property:<i>labelFormat</i>

</br>

</br>

<code>

```
let chart: Chart = new Chart({
  axes: [ { labelFormat: '{value}K' } ]
});
```

```
chart.appendTo('#chart');
```

</code>

</td>

</tr>

<tr>

<td>labelIntersectAction</td>

<td>

Property:<i>labelIntersectAction</i>

</br>

</br>

<code>

```
$("#container").ejChart({
  axes: [ { labelIntersectAction: 'trim' } ]
});
```

</code>

</td>

<td>

Property:<i>labelIntersectAction</i>

</br>

</br>

<code>

```
let chart: Chart = new Chart({
  axes: [ { labelIntersectAction: 'Trim' } ]
});
```

```
chart.appendTo('#chart');
```

</code>

</td>

```
</tr>
<tr>
<td><b>labelPosition</b></td>
<td>
<b>Property</b><i>labelPosition</i>
</br>
</br>
<code>
$( "#container" ).ejChart({
axes: [ { labelPosition: 'inside' } ]
});
</code>
</td>
<td>
<b>Property</b><i>labelPosition</i>
</br>
</br>
<code>
let chart: Chart = new Chart({
axes: [ { labelPosition: 'Inside' } ]
});
chart.appendTo('#chart');
</code>
</td>
</tr>
<tr>
<td><b>labelPlacement for category axis</b></td>
<td>
<b>Property</b><i>labelPlacement</i>
</br>
</br>
<code>
$( "#container" ).ejChart({
```

```
axes: [ { labelPlacement: 'onTicks' } ]  
});
```

```
</code>
```

```
</td>
```

```
<td>
```

```
<b>Property</b>:<i>labelPlacement</i>
```

```
</br>
```

```
</br>
```

```
<code>
```

```
let chart: Chart = new Chart({  
axes: [ { labelPlacement: 'OnTicks' } ]  
});
```

```
chart.appendTo('#chart');
```

```
</code>
```

```
</td>
```

```
</tr>
```

```
<tr>
```

```
<td><b>Axis label alignment</b></td>
```

```
<td>
```

```
<b>Property</b>:<i>alignment</i>
```

```
</br>
```

```
</br>
```

```
<code>
```

```
$("#container").ejChart({  
axes: [ { alignment: 'center' } ]  
});
```

```
</code>
```

```
</td>
```

```
<td>
```

Not Applicable

```
</td>
```

```
</tr>
```

```
<tr>
```


<td>Rotation of axis labels</td>

<td>

Property:<i>labelRotation</i>

</br>

</br>

<code>

<pre>\$("#container").ejChart({ axes: [{ labelRotation: 45 }] });</pre>

</code>

</td>

<td>

Property:<i>labelRotation</i>

</br>

</br>

<code>

<pre>let chart: Chart = new Chart({ axes: [{ labelRotation: 45 }] }); chart.appendTo('#chart');</pre>

</code>

</td>

</tr>

<tr>

<td>Log base value for logarithmic axis</td>

<td>

Property:<i>logBase</i>

</br>

</br>

<code>

<pre>\$("#container").ejChart({ axes: [{ logBase: 10 }] });</pre>

```
</code>
```

```
</td>
```

```
<td>
```

```
<b>Property</b>:<i>labelRotation</i>
```

```
</br>
```

```
</br>
```

```
<code>
```

```
let chart: Chart = new Chart({
```

```
axes: [ { logBase: 10 } ]
```

```
});
```

```
chart.appendTo('#chart');
```

```
</code>
```

```
</td>
```

```
</tr>
```

```
<tr>
```

```
<td><b>Major grid line</b></td>
```

```
<td>
```

```
<b>Property</b>:<i>majorGridLines.visible</i>
```

```
</br>
```

```
</br>
```

```
<code>
```

```
$("#container").ejChart({
```

```
axes: [ { majorGridLines: { visible: true } } ]
```

```
});
```

```
</code>
```

```
</td>
```

```
<td>
```

Not Applicable

```
</td>
```

```
</tr>
```

```
<tr>
```

```
<td><b>Width of MajorGridLines</b></td>
```

```
<td>
```

Property: *majorGridLines.width*

```
$("#container").ejChart({
  axes: [ { majorGridLines: { width: 2 } } ]
});
```

Property: *majorGridLines.width*

```
let chart: Chart = new Chart({
  axes: [ { majorGridLines: { width: 2 } } ]
});
```

```
chart.appendTo('#chart');
```

Color of MajorGridLines

Property: *majorGridLines.color*

```
$("#container").ejChart({
  axes: [ { majorGridLines: { color: 'black' } } ]
});
```

<pre> <td> Property:<i>majorGridLines.color</i> </br> </br> <code> let chart: Chart = new Chart({ axes: [{ majorGridLines: { color: 'black' } }] }); chart.appendTo('#chart'); </code> </td> </tr> <tr> <td> <td>DashArray of MajorGridLines</td> <td> Property:<i>majorGridLines.dashArray</i> </br> </br> <code> \$("#container").ejChart({ axes: [{ majorGridLines: { dashArray: 'black' } }] }); </code> </td> <td> Property:<i>majorGridLines.dashArray</i> </br> </br> <code> let chart: Chart = new Chart({ axes: [{ majorGridLines: { dashArray: 'black' } }] }); chart.appendTo('#chart'); </td></pre>	<td>DashArray of MajorGridLines</td> <td> Property:<i>majorGridLines.dashArray</i> </br> </br> <code> \$("#container").ejChart({ axes: [{ majorGridLines: { dashArray: 'black' } }] }); </code> </td> <td> Property:<i>majorGridLines.dashArray</i> </br> </br> <code> let chart: Chart = new Chart({ axes: [{ majorGridLines: { dashArray: 'black' } }] }); chart.appendTo('#chart');
---	---

</code>

</td>

</tr>

<tr>

<td>Opacity of major grid line</td>

<td>

Property:<i>majorGridLines.opacity</i>

</br>

</br>

<code>

```
$("#container").ejChart({  
  axes: [ { majorGridLines: { opacity: true } } ]  
});
```

</code>

</td>

<td>

Not Applicable

</td>

</tr>

<tr>

<td>Major Tick line</td>

<td>

Property:<i>majorTickLines.visible</i>

</br>

</br>

<code>

```
$("#container").ejChart({  
  axes: [ { majorTickLines: { visible: true } } ]  
});
```

</code>

</td>

<td>

Not Applicable

```

</td>
</tr>
<tr>
<td><b>Width of MajorTickLines</b></td>
<td>
<b>Property</b><i>majorTickLines.width</i>
</br>
</br>
<code>
$( "#container").ejChart({
axes: [ { majorTickLines: { width: 2 } } ]
});
</code>
</td>
<td>
<b>Property</b><i>majorTickLines.width</i>
</br>
</br>
<code>
let chart: Chart = new Chart({
axes: [ { majorTickLines: { width: 2 } } ]
});
chart.appendTo('#chart');
</code>
</td>
</tr>
<tr>
<td><b>Height of MajorTickLines</b></td>
<td>
<b>Property</b><i>majorTickLines.size</i>
</br>
</br>
<code>

```

```

$("#container").ejChart({
  axes: [ { majorTickLines: { size: 2 } } ]
});
</code>
</td>
<td>
<b>Property</b>:<i>majorTickLines.height</i>
</br>
</br>
<code>
let chart: Chart = new Chart({
  axes: [ { majorTickLines: { height: 2 } } ]
});
chart.appendTo('#chart');
</code>
</td>
</tr>
<tr>
<td><b>Color of MajorTickLines</b></td>
<td>
<b>Property</b>:<i>majorTickLines.color</i>
</br>
</br>
<code>
$("#container").ejChart({
  axes: [ { majorTickLines: { color: 'black' } } ]
});
</code>
</td>
<td>
<b>Property</b>:<i>majorTickLines.color</i>
</br>
</br>

```

```
<code>
```

```
let chart: Chart = new Chart({
  axes: [ { majorTickLines: { color: 'black' } } ]
});
chart.appendTo('#chart');
```

```
</code>
```

```
</td>
```

```
</tr>
```

```
<tr>
```

```
<td><b>Opacity of major Tick line</b></td>
```

```
<td>
```

```
<b>Property</b>:<i>majorTickLines.opacity</i>
```

```
</br>
```

```
</br>
```

```
<code>
```

```
$("#container").ejChart({
  axes: [ { majorTickLines: { opacity: true } } ]
});
```

```
</code>
```

```
</td>
```

```
<td>
```

Not Applicable

```
</td>
```

```
</tr>
```

```
<tr>
```

```
<td><b>maximum labels of primaryYAxis</b></td>
```

```
<td>
```

```
<b>Property</b>:<i>maximumLabels</i>
```

```
</br>
```

```
</br>
```

```
<code>
```

```
$("#container").ejChart({
  axes: [ { maximumLabels: 5 } ]
```



```

});
</code>
</td>
<td>
<b>Property</b>:<i>maximumLabels</i>
</br>
</br>
<code>
let chart: Chart = new Chart({
axes: [ { maximumLabels: 4 } ]
});
chart.appendTo('#chart');
</code>
</td>
</tr>
<tr>
<td><b>maximum labels width of primaryYAxis to trim</b></td>
<td>
<b>Property</b>:<i>maximumLabelWidth</i>
</br>
</br>
<code>
$("#container").ejChart({
axes: [ { maximumLabelWidth: 40 } ]
});
</code>
</td>
<td>
<b>Property</b>:<i>maximumLabelWidth</i>
</br>
</br>
<code>
let chart: Chart = new Chart({

```

```

axes: [ { maximumLabelWidth: 4 }}
});
chart.appendTo('#chart');
</code>
</td>
</tr>
<tr>
<td><b>minor grid line</b></td>
<td>
<b>Property</b>:<i>minorGridLines.visible</i>
</br>
</br>
<code>
$("#container").ejChart({
axes: [ { minorGridLines: { visible: true} }}
});
</code>
</td>
<td>
Not Applicable
</td>
</tr>
<tr>
<td><b>Width of minorGridLines</b></td>
<td>
<b>Property</b>:<i>minorGridLines.width</i>
</br>
</br>
<code>
$("#container").ejChart({
axes: [ { minorGridLines: { width: 2} }}
});
</code>

```

</td>

<td>

Property:<i>minorGridLines.width</i>

</br>

</br>

<code>

```
let chart: Chart = new Chart({
  axes: [ { minorGridLines: { width: 2} } ]
});
chart.appendTo('#chart');
```

</code>

</td>

</tr>

<tr>

<td>Color of minorGridLines</td>

<td>

Property:<i>minorGridLines.color</i>

</br>

</br>

<code>

```
$("#container").ejChart({
  axes: [ { minorGridLines: { color: 'black' } } ]
});
```

</code>

</td>

<td>

Property:<i>minorGridLines.color</i>

</br>

</br>

<code>

```
let chart: Chart = new Chart({
  axes: [ { minorGridLines: { color: 'black' } } ]
});
```

```

chart.appendTo('#chart');
</code>
</td>
</tr>
<tr>
<td><b>DashArray of minorGridLines</b></td>
<td>
<b>Property</b>:<i>minorGridLines.dashArray</i>
</br>
</br>
<code>
$( "#container" ).ejChart({
axes: [ { minorGridLines: { dashArray: 'black' } } ]
});
</code>
</td>
<td>
<b>Property</b>:<i>minorGridLines.dashArray</i>
</br>
</br>
<code>
let chart: Chart = new Chart({
axes: [ { minorGridLines: { dashArray: 'black' } } ]
});
chart.appendTo('#chart');
</code>
</td>
</tr>
<tr>
<td><b>Opacity of minor grid line</b></td>
<td>
<b>Property</b>:<i>minorGridLines.opacity</i>
</br>

```

</br>

<code>

```
$("#container").ejChart({  
axes: [ { minorGridLines: { opacity: true} } ]  
});
```

</code>

</td>

<td>

Not Applicable

</td>

</tr>

<tr>

<td>minor Tick line</td>

<td>

Property:<i>minorTickLines.visible</i>

</br>

</br>

<code>

```
$("#container").ejChart({  
axes: [ { minorTickLines: { visible: true} } ]  
});
```

</code>

</td>

<td>

Not Applicable

</td>

</tr>

<tr>

<td>Width of minorTickLines</td>

<td>

Property:<i>minorTickLines.width</i>

</br>

</br>

```
<code>
```

```
$("#container").ejChart({
  axes: [ { minorTickLines: { width: 2} } ]
});
```

```
</code>
```

```
</td>
```

```
<td>
```

```
<b>Property</b>:<i>minorTickLines.width</i>
```

```
</br>
```

```
</br>
```

```
<code>
```

```
let chart: Chart = new Chart({
  axes: [ { minorTickLines: { width: 2} } ]
});
chart.appendTo('#chart');
```

```
</code>
```

```
</td>
```

```
</tr>
```

```
<tr>
```

```
<td><b>Height of minorTickLines</b></td>
```

```
<td>
```

```
<b>Property</b>:<i>minorTickLines.size</i>
```

```
</br>
```

```
</br>
```

```
<code>
```

```
$("#container").ejChart({
  axes: [ { minorTickLines: { size: 2} } ]
});
```

```
</code>
```

```
</td>
```

```
<td>
```

```
<b>Property</b>:<i>minorTickLines.height</i>
```

```
</br>
```

</br>

<code>

```
let chart: Chart = new Chart({
  axes: [ { minorTickLines: { height: 2 } } ]
});
chart.appendTo('#chart');
```

</code>

</td>

</tr>

<tr>

<td>Color of minorTickLines</td>

<td>

Property:<i>minorTickLines.color</i>

</br>

</br>

<code>

```
$("#container").ejChart({
  axes: [ { minorTickLines: { color: 'black' } } ]
});
```

</code>

</td>

<td>

Property:<i>minorTickLines.color</i>

</br>

</br>

<code>

```
let chart: Chart = new Chart({
  axes: [ { minorTickLines: { color: 'black' } } ]
});
chart.appendTo('#chart');
```

</code>

</td>

</tr>

<div><tr> <td>Opacity of minor Tick line</td> <td> Property:<i>minorTickLines.opacity</i> </br> </br> <code> \$("#container").ejChart({ axes: [{ minorTickLines: { opacity: true } }] }); </code> </td> <td> Not Applicable </td> </tr> <tr> <td>Minor ticks per interval of primaryYAxis</td> <td> Property:<i>minorTicksPerInterval</i> </br> </br> <code> \$("#container").ejChart({ axes: [{ minorTicksPerInterval: 4 }] }); </code> </td> <td> Property:<i>minorTickLines.color</i> </br> </br> <code></div>	
--	--


```

let chart: Chart = new Chart({
axes: [ { minorTicksPerInterval: 4 } ]
});
chart.appendTo('#chart');
</code>
</td>
</tr>
<tr>
<td><b>name of the primaryYAxis</b></td>
<td>
<b>Property</b>:<i>name</i>
</br>
</br>
<code>
$( "#container" ).ejChart({
axes: [ { name: 'primaryYAxis' } ]
});
</code>
</td>
<td>
<b>Property</b>:<i>name</i>
</br>
</br>
<code>
let chart: Chart = new Chart({
axes: [ { name: 'primaryYAxis' } ]
});
chart.appendTo('#chart');
</code>
</td>
</tr>
<tr>
<td><b>Orientation of primaryYAxis</b></td>

```

<td>

Property:<i>orientation</i>

</br>

</br>

<code>

```
$("#container").ejChart({  
  axes: [ { orientation: 'Vertical' } ]  
});
```

</code>

</td>

<td>

Not Applicable

</td>

</tr>

<tr>

<td>Plot offset for primaryYAxis</td>

<td>

Property:<i>plotOffset</i>

</br>

</br>

<code>

```
$("#container").ejChart({  
  axes: [ { plotOffset: 0 } ]  
});
```

</code>

</td>

<td>

Property:<i>plotOffset</i>

</br>

</br>

<code>

```
let chart: Chart = new Chart({  
  axes: [ { plotOffset: 0 } ]
```

```
});  
chart.appendTo('#chart');  
</code>  
</td>  
</tr>  
<tr>  
<td><b>minimum for primaryYAxis</b></td>  
<td>  
<b>Property</b>:<i>range.minimum</i>  
</br>  
</br>  
<code>  
$("#container").ejChart({  
axes: [ { range: { minimum: 10 }}]  
});  
</code>  
</td>  
<td>  
<b>Property</b>:<i>minimum</i>  
</br>  
</br>  
<code>  
let chart: Chart = new Chart({  
axes: [ { minimum: 23 }]  
});  
chart.appendTo('#chart');  
</code>  
</td>  
</tr>  
<tr>  
<td><b>maximum for primaryYAxis</b></td>  
<td>  
<b>Property</b>:<i>range.maximum</i>
```

</br>

</br>

<code>

```
$("#container").ejChart({
  axes: [ { range: { maximum: 10 }}]
});
```

</code>

</td>

<td>

Property:<i>maximum</i>

</br>

</br>

<code>

```
let chart: Chart = new Chart({
  axes: [ { maximum: 23 }]
});
```

```
chart.appendTo('#chart');
```

</code>

</td>

</tr>

<tr>

<td>interval for primaryYAxis</td>

<td>

Property:<i>range.interval</i>

</br>

</br>

<code>

```
$("#container").ejChart({
  axes: [ { range: { interval: 1 }}]
});
```

</code>

</td>

<td>

Property:*interval*

```
let chart: Chart = new Chart({
```

```
  axes: [ { interval: 2 }]
```

```
});
```

```
chart.appendTo('#chart');
```

RangePadding for primaryYAxis

Property:*rangePadding*

```
$("#container").ejChart({
```

```
  axes: [ { rangePadding: 'None' }]
```

```
});
```

Property:*rangePadding*

```
let chart: Chart = new Chart({
```

```
  axes: [ { rangePadding: 'None' }]
```

```
});
```

```
chart.appendTo('#chart');
```

```
</td>
</tr>
<tr>
<td><b>Rounding Places in primaryYAxis</b></td>
<td>
<b>Property</b><i>roundingPlaces </i>
</br>
</br>
<code>
$( "#container" ).ejChart({
axes: [ { roundingPlaces: 3 } ]
});
</code>
</td>
<td>
<b>Property</b><i>labelFormat</i>
</br>
</br>
<code>
let chart: Chart = new Chart({
axes: [ { labelFormat: 'n3' } ]
});
chart.appendTo('#chart');
</code>
</td>
</tr>
<tr>
<td><b>ScrollBar settings of primaryYAxis</b></td>
<td>
<b>Property</b><i>scrollbarSettings </i>
</br>
</br>
<code>
```

```
$("#container").ejChart({  
  axes: [ { scrollbarSettings : { } } ]  
});
```

</code>

</td>

<td>

Not Applicable

</td>

</tr>

<tr>

<td>TickPosition in primaryYAxis</td>

<td>

Property:<i>tickLinesPosition</i>

</br>

</br>

<code>

```
$("#container").ejChart({  
  axes: [ { tickLinesPosition: 'Inside' } ]  
});
```

</code>

</td>

<td>

Property:<i>tickPosition</i>

</br>

</br>

<code>

```
let chart: Chart = new Chart({  
  axes: [ { tickPosition: 'Inside' } ]  
});
```

```
chart.appendTo('#chart');
```

</code>

</td>

</tr>

```
<tr>
<td><b>valueType of primaryYAxis</b></td>
<td>
<b>Property</b>:<i>valueType</i>
</br>
</br>
<code>
$( "#container" ).ejChart({
axes: [ { valueType: 'DateTime' } ]
});
</code>
</td>
<td>
<b>Property</b>:<i>valueType</i>
</br>
</br>
<code>
let chart: Chart = new Chart({
axes: [ { valueType: 'DateTime' } ]
});
chart.appendTo('#chart');
</code>
</td>
</tr>
<tr>
<td><b>visible of primaryYAxis</b></td>
<td>
<b>Property</b>:<i>visible</i>
</br>
</br>
<code>
$( "#container" ).ejChart({
axes: [ { visible: true } ]
```



```
});
</code>
</td>
<td>
<b>Property</b>:<i>visible</i>
</br>
</br>
<code>
let chart: Chart = new Chart({
axes: [ { visible: true } ]
});
chart.appendTo('#chart');
</code>
</td>
</tr>
<tr>
<td><b>zoomFactor of primaryYAxis</b></td>
<td>
<b>Property</b>:<i>zoomFactor</i>
</br>
</br>
<code>
$("#container").ejChart({
axes: [ { zoomFactor: 0.3 } ]
});
</code>
</td>
<td>
<b>Property</b>:<i>zoomFactor</i>
</br>
</br>
<code>
let chart: Chart = new Chart({
```

```

axes: [ { zoomFactor: 0.3 }
});
chart.appendTo('#chart');
</code>
</td>
</tr>
<tr>
<td><b>zoomPosition of primaryYAxis</b></td>
<td>
<b>Property</b>:<i>zoomPosition</i>
</br>
</br>
<code>
$( "#container" ).ejChart({
axes: [ { zoomPosition: 0.3 }
});
</code>
</td>
<td>
<b>Property</b>:<i>zoomPosition</i>
</br>
</br>
<code>
let chart: Chart = new Chart({
axes: [ { zoomPosition: 0.3 }
});
chart.appendTo('#chart');
</code>
</td>
</tr>
<tr>
<td><b>labelBorder of primaryYAxis</b></td>
<td>

```

```
<b>Property</b><i>labelBorder</i>
</br>
</br>
<code>
$("#container").ejChart({
axes: [ { labelBorder: { color: 'red', width: 2 } }]
});
</code>
</td>
<td>
<b>Property</b><i>border</i>
</br>
</br>
<code>
let chart: Chart = new Chart({
axes: [ { border: { color: 'red', width: 3 } }]
});
chart.appendTo('#chart');
</code>
</td>
</tr>
<tr>
<td><b>title of primaryYAxis</b></td>
<td>
<b>Property</b><i>title.text</i>
</br>
</br>
<code>
$("#container").ejChart({
axes: [ { title: { text: 'Chart title' } }]
});
</code>
</td>
```

```
<td>
<b>Property</b>:<i>title</i>
</br>
</br>
<code>
let chart: Chart = new Chart({
axes: [ { title: 'Chart title' } ]
});
chart.appendTo('#chart');
</code>
</td>
</tr>
<tr>
<td><b>StripLine of primaryYAxis</b></td>
<td>
<b>Property</b>:<i>stripLine</i>
</br>
</br>
<code>
$("#container").ejChart({
axes: [ { stripLine: [] } ]
});
</code>
</td>
<td>
<b>Property</b>:<i>stripLines</i>
</br>
</br>
<code>
let chart: Chart = new Chart({
axes: [ { stripLines: [] } ]
});
chart.appendTo('#chart');
```

```
</code>
</td>
</tr>
<tr>
<td><b>Multilevel labels of axes</b></td>
<td>
<b>Property</b>:<i>multiLevelLabels</i>
<br>
<br>
<code>
$( "#container" ).ejChart({
axes: [ { multiLevelLabels: [] } ]
});
</code>
</td>
<td>
<b>Property</b>:<i>multiLevelLabels</i>
<br>
<br>
<code>
let chart: Chart = new Chart({
axes: [ { stripLines: [] } ]
});
chart.appendTo('#chart');
</code>
</td>
</tr>
<tr>
<td><b>skeleton for an axes</b></td>
<td>
Not Applicable
</td>
<td>
```

Property<i>skeleton</i>

</br>

</br>

<code>

let chart: Chart = new Chart({

axes: [{ skeleton: 'yMd' } }

});

chart.appendTo('#chart');

</code>

</td>

</tr>

<tr>

<td>skeleton type for an axes</td>

<td>

Not Applicable

</td>

<td>

Property<i>skeletonType</i>

</br>

</br>

<code>

let chart: Chart = new Chart({

axes: [{ skeletonType: 'DateTime' } }

});

chart.appendTo('#chart');

</code>

</td>

</tr>

## Rows		
Behaviour	API in Essential JS 1	API in Essential JS 2
rows in chart	Property:rowDefinitions\$("#chart").ejChart({ rowDefinitions: []});	Property:rowslet chart: Chart = new Chart({ rows: []});chart.appendTo('#chart');

unit	Property:unit \$("#container").ejChart({ rowDefinitions :[{unit : "percentage"}]});	Not Applicable
height of rows in chart	Property:rowHeight\$("#chart").ejChart({ rowDefinitions: [{ rowHeight: '50%'}]});	Property:heightlet chart: Chart = new Chart({ rows: [{ height: '300'}]});chart.appendTo('#chart');
Line customization	Property:lineColor, lineWidth\$("#chart").ejChart({ rowDefinitions: [{ rowHeight: '50%', lineColor: 'brown', lineWidth: 2}]});	Property:borderlet chart: Chart = new Chart({ rows: [{ height: '300', border: { width: 2, color: 'brown'}]}]});chart.appendTo('#chart');
## Series		
Behaviour	API in Essential JS 1	API in Essential JS 2
bearFillColor	Property:bearFillColor\$("#chart").ejChart({ series: [{bearFillColor: 'red'}]});	Property:bearFillColorlet chart: Chart = new Chart({ series: [{bearFillColor: 'red'}]});chart.appendTo('#chart');
Border	Property:border\$("#chart").ejChart({ series: [{ border: { color: 'red', width: 2, dashArray: '10, 5' } }]});	Property:rowslet chart: Chart = new Chart({ series: [{ border: { color: 'red', width: 2 } }]});chart.appendTo('#chart');
BoxPlotMode	Property:boxPlotMode\$("#chart").ejChart({ series: [{ boxPlotMode: 'inclusive' }]});	Property:rowslet chart: Chart = new Chart({ series: [{ boxPlotMode: 'Inclusive' }]});chart.appendTo('#chart');
Minimum radius of Bubble series	Property:bubbleOptions.minRadius\$("#chart").ejChart({ series: [{ bubbleOptions: { minRadius: 2 } }]});	Property:minRadiuslet chart: Chart = new Chart({ series: [{ minRadius: 2 }]});chart.appendTo('#chart');
Maximum radius of Bubble series	Property:bubbleOptions.maxRadius\$("#chart").ejChart({ series: [{ bubbleOptions: { maxRadius: 10 } }]});	Property:maxRadiuslet chart: Chart = new Chart({ series: [{ maxRadius: 2 }]});chart.appendTo('#chart');
bullFillColor	Property:bullFillColor\$("#chart").ejChart({ series: [{bullFillColor: 'red'}]});	Property:bullFillColorlet chart: Chart = new Chart({ series: [{bullFillColor: 'red' }]});

		<code>});});chart.appendTo('#chart');</code>
Cardinal spline tension for spline series	<code>Property:cardinalSplineTension\$("#chart").ejChart({ series: [{ cardinalSplineTension: 0.5 }]});</code>	<code>Property:cardinalSplineTensionlet chart: Chart = new Chart({ series: [{ cardinalSplineTension: 0.5 }]});chart.appendTo('#chart');</code>
Column Width for rectangle series	<code>Property:columnWidth\$("#chart").ejChart({ series: [{ columnWidth: 0.5 }]});</code>	<code>Property:columnWidthlet chart: Chart = new Chart({ series: [{ columnWidth: 0.5 }]});chart.appendTo('#chart');</code>
Column spacing for rectangle series	<code>Property:columnSpacing\$("#chart").ejChart({ series: [{ columnSpacing: 0.5 }]});</code>	<code>Property:columnSpacinglet chart: Chart = new Chart({ series: [{ columnSpacing: 0.5 }]});chart.appendTo('#chart');</code>
Topleft radius for rectangle series	<code>Property:cornerRadius.topLeft\$("#chart").ejChart({ series: [{ topLeft: 0 }]});</code>	<code>Property:cornerRadius.topLeftlet chart: Chart = new Chart({ series: [{ topLeft: 0 }]});chart.appendTo('#chart');</code>
topRight radius for rectangle series	<code>Property:cornerRadius.topRight\$("#chart").ejChart({ series: [{ topRight: 0 }]});</code>	<code>Property:cornerRadius.topRightlet chart: Chart = new Chart({ series: [{ topRight: 0 }]});chart.appendTo('#chart');</code>
bottomRight radius for rectangle series	<code>Property:cornerRadius.bottomRight\$("#chart").ejChart({ series: [{ bottomRight: 0 }]});</code>	<code>Property:cornerRadius.bottomRightlet chart: Chart = new Chart({ series: [{ bottomRight: 0 }]});chart.appendTo('#chart');</code>
bottomLeft radius for rectangle series	<code>Property:cornerRadius.bottomLeft\$("#chart").ejChart({ series: [{ bottomLeft: 0 }]});</code>	<code>Property:cornerRadius.bottomLeftlet chart: Chart = new Chart({ series: [{ bottomLeft: 0 }]});chart.appendTo('#chart');</code>
DashArray property	<code>Property:dashArray\$("#chart").ejChart({ series: [{ dashArray: '10, 5' }]});</code>	<code>Property:dashArraylet chart: Chart = new Chart({ series: [{ dashArray: '10, 5' }]});</code>

		<code>});});chart.appendTo('#chart');</code>
Data Source for series	<code>Property:dataSource\$("#chart").ejChart({ series: [{ dataSource: [] }]});</code>	<code>Property:dashArraylet chart: Chart = new Chart({ series: [{ dataSource: [] }]});chart.appendTo('#chart');</code>
Draw type for Polar series	<code>Property:drawType\$("#chart").ejChart({ series: [{ drawType: 'Line' }]});</code>	<code>Property:drawTypelet chart: Chart = new Chart({ series: [{ drawType: 'Line' }]});chart.appendTo('#chart');</code>
Empty Point Settings for series	<code>Property:emptyPointSettings.visible\$("#chart").ejChart({ series: [{ emptyPointSettings: { visible: false } }]});</code>	Not Applicable
Empty Point Display mode	<code>Property:emptyPointSettings.displayMode\$("#chart").ejChart({ series: [{ displayMode: 'gap' }]});</code>	<code>Property:emptyPointSettings.displayModelet chart: Chart = new Chart({ series: [{ displayMode: 'Average' }]});chart.appendTo('#chart');</code>
Empty Point color	<code>Property:emptyPointSettings.color\$("#chart").ejChart({ series: [{ color: 'red' }]});</code>	<code>Property:emptyPointSettings.filllet chart: Chart = new Chart({ series: [{ fill: 'red' }]});chart.appendTo('#chart');</code>
Empty Point Border	<code>Property:emptyPointSettings.border\$("#chart").ejChart({ series: [{ emptyPointSettings: { color: 'red', width: 2 } }]});</code>	<code>Property:filllet chart: Chart = new Chart({ series: [{ emptyPointSettings: { color: 'red', width: 2 } }]});chart.appendTo('#chart');</code>
Enable animation for series	<code>Property:enableAnimation\$("#chart").ejChart({ series: [{ enableAnimation: true }]});</code>	<code>Property:animation.enablelet chart: Chart = new Chart({ series: [animation: { enable: false }]});chart.appendTo('#chart');</code>
Animation duration for series	<code>Property:animationDuration\$("#chart").ejChart({ series: [{ animationDuration: 1000 }]});</code>	<code>Property:animation.durationlet chart: Chart = new Chart({ series: [animation: { duration: 1000 }]});chart.appendTo('#chart');</code>

Animation delay for series	Not Applicable	Property:animation.durationlet chart: Chart = new Chart({ series: [animation: { delay: 100 }] }); chart.appendTo('#chart');
Drag settings for series	Property:dragSettings\$("#chart").ejChart({ series: [{ dragSettings: { mode: 'X' } }] });	Not Applicable
Errorbar settings for series	Property:errorBarSettings\$("#chart").ejChart({ series: [{ errorBarSettings: { } }] });	Property:errorBarSettingslet chart: Chart = new Chart({ series: [{errorBarSettings: { } }] });
Closed series	Property:isClosed\$("#chart").ejChart({ series: [{ isClosed: true }] });	Property:isClosedlet chart: Chart = new Chart({ series: [{ isClosed: true }] });
Stacking Property for series	Property:isStacking\$("#chart").ejChart({ series: [{ isStacking: true }] });	Not Applicable
Line cap for series	Property:lineCap\$("#chart").ejChart({ series: [{ lineCap: 'butt' }] });	Not Applicable
Line join for series	Property:lineJoin\$("#chart").ejChart({ series: [{ lineJoin: 'round' }] });	Not Applicable
Opacity for series	Property:opacity\$("#chart").ejChart({ series: [{ opacity: 0.7 }] });	Property:opacitylet chart: Chart = new Chart({ series: [{ opacity: 0.7 }] });
Outlier settings of series	Property:outLierSettings\$("#chart").ejChart({ series: [{ outLierSettings: { shape: 'rectangle' , size: { height: 30, width: 20 } } }] });	Not Applicable
Palette	Property:palette\$("#chart").ejChart({ series: [{ palette: "ColorFieldName" }] });	Property:pointColorMappinglet chart: Chart = new Chart({ series: [{ pointColorMapping: 'color' }] }); chart.appendTo('#chart');
Positive fill for waterfall series	Property:positiveFill\$("#chart").ejChart({ series: [{ positiveFill: "red" }] });	Property:pointColorMappinglet chart: Chart = new Chart({ series: [{ pointColorMapping: 'color' }] }); chart.appendTo('#chart');

Show average value in box and whisker series	Property:showMedian\$("#chart").ejChart({ series: [{ showMedian: true }];});	Property:pointColorMappinglet chart: Chart = new Chart({ series: [{ showMean: false }];});chart.appendTo('#chart');
To group the series of stacking collection.	Property:stackingGroup\$("#chart").ejChart({ series: [{ stackingGroup: 'group' }];});	Property:stackingGrouplet chart: Chart = new Chart({ series: [{ stackingGroup: 'group' }];});chart.appendTo('#chart');
Specifies the type of the series to render in chart.	Property:type\$("#chart").ejChart({ series: [{ type: 'Line' }];});	Property:typetlet chart: Chart = new Chart({ series: [{ type: 'Line' }];});chart.appendTo('#chart');
Defines the visibility of the series.	Property:visibility\$("#chart").ejChart({ series: [{ visibility: true }];});	Property:visiblelet chart: Chart = new Chart({ series: [{ visible: true }];});chart.appendTo('#chart');
Enables or disables the visibility of legend item.	Property:visibleOnLegend \$("#chart").ejChart({ series: [{ visibleOnLegend : true }];});	Property:toggleVisibilitylet chart: Chart = new Chart({ legendSettings: [{ toggleVisibility: true }];});chart.appendTo('#chart');
Specifies the different types of spline curve.	Property:splineType \$("#chart").ejChart({ series: [{ splineType : 'Natural' }];});	Property:splineTypetlet chart: Chart = new Chart({ legendSettings: [{ splineType: 'Natural' }];});chart.appendTo('#chart');
Specifies the name of the x-axis that has to be associated with this series. Add an axis instance with this name to axes collection.	Property:xAxisName\$("#chart").ejChart({ series: [{ xAxisName : 'secondaryXAxis' }];});	Property:xAxisNamelet chart: Chart = new Chart({ series: [{ xAxisName: 'secondaryXAxis' }];});chart.appendTo('#chart');

Name of the property in the datasource that contains x value for the series.	Property:xName\$("#chart").ejChart({ series: [{ xName : 'x' }]});	Property:xNamelet chart: Chart = new Chart({ series: [{ xName: 'x' }]});chart.appendTo('#chart');
Specifies the name of the y-axis that has to be associated with this series. Add an axis instance with this name to axes collection.	Property:yAxisName\$("#chart").ejChart({ series: [{ yAxisName : 'secondaryYAxis' }]});	Property:yAxisNamelet chart: Chart = new Chart({ series: [{ yAxisName: 'secondaryYAxis' }]});chart.appendTo('#chart');
Name of the property in the datasource that contains y value for the series.	Property:yName\$("#chart").ejChart({ series: [{ yName : 'y' }]});	Property:yNamelet chart: Chart = new Chart({ series: [{ yName: 'y' }]});chart.appendTo('#chart');
Name of the property in the datasource that contains high value for the series.	Property:high\$("#chart").ejChart({ series: [{ high : 'y' }]});	Property:highlet chart: Chart = new Chart({ series: [{ high: 'y' }]});chart.appendTo('#chart');
Name of the property in the datasource that contains low value for the series.	Property:low\$("#chart").ejChart({ series: [{ low : 'y' }]});	Property:lowlet chart: Chart = new Chart({ series: [{ low: 'y' }]});chart.appendTo('#chart');
Name of the property in the datasource that contains	Property:close\$("#chart").ejChart({ series: [{ close : 'y' }]});	Property:closelet chart: Chart = new Chart({ series: [{ close: 'y' }]});chart.appendTo('#chart');

close value for the series.		
Name of the property in the datasource that contains open value for the series.	Property:open\$("#chart").ejChart({ series: [{ open : 'y' }]});	Property:openlet chart: Chart = new Chart({ series: [{ open: 'y' }]});chart.appendTo('#chart');
Option to add trendlines to chart.	Property:trendLines\$("#chart").ejChart({ series: [{ trendLines : [{}]}]});	Property:trendLineslet chart: Chart = new Chart({ series: [{ trendLines : [{}]}]});chart.appendTo('#chart');
Options for customizing the appearance of the series or data point while highlighting.	Property:highlightSettings\$("#chart").ejChart({ series: [{ highlightSettings : {} }]});	Not applicable.
Options for customizing the appearance of the series/data point on selection.	Property:selectionSettings\$("#chart").ejChart({ series: [{ selectionSettings : {} }]});	Not applicable.
## marker		
visibility of marker	Property:visible\$("#chart").ejChart({ series: [{ marker: { visible: true } }]});	Property:visiblelet chart: Chart = new Chart({ series: [{ marker: { visible: false } }]});chart.appendTo('#chart');
Fill for marker	Property:fill\$("#chart").ejChart({ series: [{ marker: { fill : 'red' } }]});	Property:visiblelet chart: Chart = new Chart({ series: [{ marker: { fill : 'red' } }]});chart.appendTo('#chart');

Opacity for marker	Property:opacity\$("#chart").ejChart({ series: [{ marker: { opacity : 0.5 } }]});	Property:opacitylet chart: Chart = new Chart({ series: [{ marker: { opacity : 0.5 } }]});chart.appendTo('#chart');
Shape of marker	Property:shape\$("#chart").ejChart({ series: [{ marker: { shape : 'Circle' } }]});	Property:shapelet chart: Chart = new Chart({ series: [{ marker: { shape : 'Triangle' } }]});chart.appendTo('#chart');
ImageUrl of marker	Property:imageUrl\$("#chart").ejChart({ series: [{ marker: { imageUrl : '' } }]});	Property:imageUrllet chart: Chart = new Chart({ series: [{ marker: { imageUrl : '' } }]});chart.appendTo('#chart');
Border of marker	Property:border\$("#chart").ejChart({ series: [{ marker: { border : { width: 2, color: 'red' } } }]});	Property:shapelet chart: Chart = new Chart({ series: [{ marker: { border : { width: 2, color: 'red' } } }]});chart.appendTo('#chart');
Height of marker	Property:size.height\$("#chart").ejChart({ series: [{ marker: { size: { height: 30 } } }]});	Property:heightlet chart: Chart = new Chart({ series: [{ marker: { height: 25 } }]});chart.appendTo('#chart');
Width of marker	Property:size.width\$("#chart").ejChart({ series: [{ marker: { size: { width: 30 } } }]});	Property:widthlet chart: Chart = new Chart({ series: [{ marker: { width: 25 } }]});chart.appendTo('#chart');
Width of marker	Property:size.width\$("#chart").ejChart({ series: [{ marker: { size: { width: 30 } } }]});	Property:widthlet chart: Chart = new Chart({ series: [{ marker: { width: 25 } }]});chart.appendTo('#chart');
DataLabelSettings of marker	Property:marker.dataLabel\$("#chart").ejChart({ series: [{ marker: { dataLabel: { } } }]});	Property:marker.dataLabellet chart: Chart = new Chart({ series: [{ marker: { dataLabel: { } } }]});

		<code>}};});chart.appendTo('#chart');</code>
Visibility of dataLabel	<code>Property:dataLabel.visible\$("#chart").ejChart({ series: [{ marker: {dataLabel: { visible: true } } }]});</code>	<code>Property:dataLabel.visiblelet chart: Chart = new Chart({ series: [{ marker: { dataLabel: { visible: true } } }]});chart.appendTo('#chart');</code>
Text mapping name of dataLabel	<code>Property:dataLabel.textMappingName\$("#chart").ejChart({ series: [{ marker: {dataLabel: { textMappingName: '' } } }]});</code>	<code>Property:dataLabel.namelet chart: Chart = new Chart({ series: [{ marker: { dataLabel: { name: '' } } }]});chart.appendTo('#chart');</code>
Fill color of data label	<code>Property:dataLabel.fill\$("#chart").ejChart({ series: [{ marker: {dataLabel: { fill: 'pink' } } }]});</code>	<code>Property:dataLabel.filllet chart: Chart = new Chart({ series: [{ marker: { dataLabel: { fill: 'pink' } } }]});chart.appendTo('#chart');</code>
Opacity of data label	<code>Property:dataLabel.opacity\$("#chart").ejChart({ series: [{ marker: {dataLabel: { opacity: 0.6 } } }]});</code>	<code>Property:dataLabel.filllet chart: Chart = new Chart({ series: [{ marker: { dataLabel: { opacity: 0.4 } } }]});chart.appendTo('#chart');</code>
Text position of data label	<code>Property:dataLabel.textPosition\$("#chart").ejChart({ series: [{ marker: {dataLabel: { textPosition: 'middle' } } }]});</code>	<code>Property:dataLabel.positionlet chart: Chart = new Chart({ series: [{ marker: { dataLabel: { position: 'Top' } } }]});chart.appendTo('#chart');</code>
Alignment of data label	<code>Property:dataLabel.verticalAlignment\$("#chart").ejChart({ series: [{ marker: {dataLabel: { verticalAlignment: 'near' } } }]});</code>	<code>Property:dataLabel.alignmentlet chart: Chart = new Chart({ series: [{ marker: { dataLabel: { alignment: 'Near' } } }]});chart.appendTo('#chart');</code>
Border of data label	<code>Property:dataLabel.border\$("#chart").ejChart({ series: [{ marker: {dataLabel: { border: { color: 'blue', width: 2, opacity: 0.4 } } }]});</code>	<code>Property:dataLabel.alignmentlet chart: Chart = new Chart({ series: [{ marker: { dataLabel: { border: { color: 'blue', width: 2 } } }]});</code>

		<code>}};});chart.appendTo('#chart');</code>
Offset for data label	<code>Property:dataLabel.offset\$("#chart").ejChart({ series: [{ marker: {dataLabel: { offset: { x: 5, y: 6 } } } }]});</code>	Not Applicable
Margin of data label	<code>Property:dataLabel.border\$("#chart").ejChart({ series: [{ marker: {dataLabel: { margin: { top: 10, bottom: 10, left: 10, right: 10 } } } }]});</code>	<code>Property:dataLabel.marginlet chart: Chart = new Chart({ series: [{ marker: { dataLabel: { margin: { top: 10, bottom: 10, left: 10, right: 10 } } } }]});chart.appendTo('#chart');</code>
Font of data label	<code>Property:dataLabel.border\$("#chart").ejChart({ series: [{ marker: {dataLabel: { font: { fontFamily: 'SegoeUI', fontStyle: 'italic', fontWeight: '600', opacity: 0.5, size: 12, color: 'red' } } } }]});</code>	<code>Property:dataLabel.marginlet chart: Chart = new Chart({ series: [{ marker: {dataLabel: { font: { fontFamily: 'SegoeUI', fontStyle: 'italic', fontWeight: '600', opacity: 0.5, size: 12, color: 'red' } } } }]});chart.appendTo('#chart');</code>
HTML template in dataLabel	<code>Property:dataLabel.template\$("#chart").ejChart({ series: [{ marker: {dataLabel: { template: 'Chart' } } }]});</code>	<code>Property:dataLabel.templatelet chart: Chart = new Chart({ series: [{ marker: {dataLabel: { template: 'Chart' } } }]});chart.appendTo('#chart');</code>
Rounded corner radius X	Not Applicable	<code>Property:dataLabel.rxlet chart: Chart = new Chart({ series: [{ marker: {dataLabel: { rx: 10 } } }]});chart.appendTo('#chart');</code>
Rounded corner radius Y	Not Applicable	<code>Property:dataLabel.rylet chart: Chart = new Chart({ series: [{ marker: {dataLabel: { ry: 10 } } }]});chart.appendTo('#chart');</code>

Maximum Label width for data label	Property:dataLabel.maximumLabelWidth\$("#chart").ejChart({ series: [{ marker: {dataLabel: { maximumLabelWidth: 20}} } }]);	Not Applicable
Enable wrapping of text for data label	Property:dataLabel.enableWrap\$("#chart").ejChart({ series: [{ marker: {dataLabel: { enableWrap: true }} } }]);	Not Applicable
To show contrast color for data label	Property:dataLabel.showContrastColor\$("#chart").ejChart({ series: [{ marker: {dataLabel: { showContrastColor: true }} } }]);	Not Applicable
To show edge label for data label	Property:dataLabel.showEdgeLabels\$("#chart").ejChart({ series: [{ marker: {dataLabel: { showEdgeLabels: true }} } }]);	Not Applicable
## TrendLines		
Behavior	API in Essential JS 1	API in Essential JS 2
Trendlines settings	Property:series.trendLines\$("#chart").ejChart({ series: [{ trendLines: []}]});	Property:series.trendLineslet chart: Chart = new Chart({ series: [{ trendLines: []}]});chart.appendTo('#chart');
Visibility of trendline	Property:trendLines.visibility\$("#chart").ejChart({ series: [{ trendLines: [visibility: true]}]});	Not applicable
Type of trendline	Property:trendLines.type\$("#chart").ejChart({ series: [{ trendLines: [type: 'linear']}]});	Property:trendLines.typelet chart: Chart = new Chart({ series: [{ trendLines: [type: 'Polynomial']}]});chart.appendTo('#chart');
Name of trendline	Property:trendLines.name\$("#chart").ejChart({ series: [{ trendLines: [name: 'trendLine']}]});	Property:trendLines.namelet chart: Chart = new Chart({ series: [{ trendLines: [name: 'trendLine']}]});chart.appendTo('#chart');
Period of trendline	Property:trendLines.period\$("#chart").ejChart({ series: [{ trendLines: [period: 45]}]});	Property:trendLines.periodlet chart: Chart = new Chart({ series: [{ trendLines: [period: 45]}]});chart.appendTo('#chart');
Polynomial order for	Property:trendLines.polynomialOrder\$("#chart").ejChart({ series: [{ trendLines: [polynomialOrder: 3]}]});	Property:trendLines.polynomialOrderlet chart: Chart = new Chart({ series: [{ trendLines: [

Polynomial type trendLines		polynomialOrder: 3]]]]);chart.appendTo('#chart');
Backward forecast for trendLines	Property:trendLines.backwardforecast\$("#chart").ejChart({ series: [{ trendLines: [backwardforecast: 3]}]]]);	Property:trendLines.backwardforecastlet chart: Chart = new Chart({ series: [{ trendLines: [backwardforecast: 3]}]]]);chart.appendTo('#chart');
Forward forecast for trendLines	Property:trendLines.forwardForecast\$("#chart").ejChart({ series: [{ trendLines: [forwardForecast: 3]}]]]);	Property:trendLines.forwardForecastlet chart: Chart = new Chart({ series: [{ trendLines: [forwardForecast: 3]}]]]);chart.appendTo('#chart');
Fill for trendLines	Property:trendLines.fill\$("#chart").ejChart({ series: [{ trendLines: [fill: '#EEFFCC']}]]]);	Property:trendLines.filllet chart: Chart = new Chart({ series: [{ trendLines: [fill: 'EEFFCC']}]]]);chart.appendTo('#chart');
Width for trendLines	Property:trendLines.width\$("#chart").ejChart({ series: [{ trendLines: [width: 2]}]]]);	Property:trendLines.widthlet chart: Chart = new Chart({ series: [{ trendLines: [width: 2]}]]]);chart.appendTo('#chart');
Intercept value for trendLines	Property:trendLines.intercept\$("#chart").ejChart({ series: [{ trendLines: [intercept: 2]}]]]);	Property:trendLines.interceptlet chart: Chart = new Chart({ series: [{ trendLines: [intercept: 2]}]]]);chart.appendTo('#chart');
Legend shape for trendLines	Not Applicable	Property:trendLines.legendShapelet chart: Chart = new Chart({ series: [{ trendLines: [legendShape: 'Rectangle']}]]]);chart.appendTo('#chart');
Animation settings for trendLines	Not Applicable	Property:trendLines.animationlet chart: Chart = new Chart({ series: [{ trendLines: [animation: { enable: true }]}]]]);chart.appendTo('#chart');
Marker settings	Not Applicable	Property:trendLines.markerlet chart: Chart = new Chart({ series: [{

for trendLines		trendLines: [{marker: { visible: true }}}]);chart.appendTo('#chart');
Tooltip for trendLines	Property:trendLines.tooltip\$("#chart").ejChart({ series: [{ trendLines: [{ tooltip: { } }]}]})	Property:trendLines.enableTooltiplet chart: Chart = new Chart({ series: [{ trendLines: [{enableTooltip: true }]}]);chart.appendTo('#chart');
DashArray for trendLines	Property:trendLines.dashArray\$("#chart").ejChart({ series: [{ trendLines: [{ dashArray: '10, 5' }]}]})	Not Applicable.
Visible on legend for trendLines	Property:trendLines.visibleOnLegend\$("#chart").ejChart({ series: [{ trendLines: [{ visibleOnLegend: true }]}]})	Not Applicable.
## StripLines		
Behaviour	API in Essential JS 1	API in Essential JS 2
Default behaviour for striplines	Property:primaryXAxis.stripLines\$("#chart").ejChart({ primaryXAxis: { stripLines: [{ visible: true }]}});	Property:primaryXAxis.stripLineslet chart: Chart = new Chart({ primaryXAxis: { stripLines: [{ visible: true }]}});chart.appendTo('#chart');
border for stripline	Property:stripLines.borderColor\$("#chart").ejChart({ primaryXAxis: { stripLines: [{ borderColor: 'pink' }]}});	Property:stripLines.borderlet chart: Chart = new Chart({ primaryXAxis: { stripLines: [{ border: { color: 'red', width: 2 } }]}});chart.appendTo('#chart');
Background color for stripline	Property:stripLines.color\$("#chart").ejChart({ primaryXAxis: { stripLines: [{ color: 'pink' }]}});	Property:stripLines.borderlet chart: Chart = new Chart({ primaryXAxis: { stripLines: [{ color: 'red' }]}});chart.appendTo('#chart');
Start value for stripline	Property:stripLines.start\$("#chart").ejChart({ primaryXAxis: { stripLines: [{ start: 10 }]}});	Property:stripLines.startlet chart: Chart = new Chart({ primaryXAxis: { stripLines: [{ start: 5 }]}});chart.appendTo('#chart');

End value for stripline	Property:stripLines.end\$("#chart").ejChart({ primaryXAxis: { stripLines: [{ end: 10 }] } });	Property:stripLines.endlet chart: Chart = new Chart({ primaryXAxis: { stripLines: [{ end: 5 }] } }); chart.appendTo('#chart');
Startfrom Axis for stripline	Property:stripLines.startFromAxis\$("#chart").ejChart({ primaryXAxis: { stripLines: [{ startFromAxis: true }] } });	Property:stripLines.startFromAxislet chart: Chart = new Chart({ primaryXAxis: { stripLines: [{ startFromAxis: true }] } }); chart.appendTo('#chart');
Text in stripline	Property:stripLines.text\$("#chart").ejChart({ primaryXAxis: { stripLines: [{ text: 'StripLine; }] } });	Property:stripLines.textlet chart: Chart = new Chart({ primaryXAxis: { stripLines: [{ text: 'stripline' }] } }); chart.appendTo('#chart');
Text alignment in stripline	Property:stripLines.textAlignment\$("#chart").ejChart({ primaryXAxis: { stripLines: [{ textAlignment: 'Far; }] } });	Property:stripLines.horizontalAlignmentlet chart: Chart = new Chart({ primaryXAxis: { stripLines: [{ horizontalAlignment: 'Far' }] } }); chart.appendTo('#chart');
Vertical Text alignment in stripline	Not Applicable	Property:stripLines.verticalAlignmentlet chart: Chart = new Chart({ primaryXAxis: { stripLines: [{ verticalAlignment: 'Far' }] } }); chart.appendTo('#chart');
Size of stripline	Property:stripLines.width\$("#chart").ejChart({ primaryXAxis: { stripLines: [{ width: 10; }] } });	Property:stripLines.sizelet chart: Chart = new Chart({ primaryXAxis: { stripLines: [{ size: 10 }] } }); chart.appendTo('#chart');
ZIndex of stripline	Property:stripLines.zIndex\$("#chart").ejChart({ primaryXAxis: { stripLines: [{ zIndex: 'Behind' }] } });	Property:stripLines.sizelet chart: Chart = new Chart({ primaryXAxis: { stripLines: [{ zIndex: 'Behind' }] } }); chart.appendTo('#chart');
Font style of stripline	Property:stripLines.fontStyle\$("#chart").ejChart({ primaryXAxis: { stripLines: [{ fontStyle: {} }] } });	Property:stripLines.textStylelet chart: Chart = new Chart({ primaryXAxis: { stripLines: [{ textStyle: {} }] } }); chart.appendTo('#chart');
## Multilevel Labels		
Behaviour	API in Essential JS 1	API in Essential JS 2

Default behaviour for multilevel Labels	Property:primaryXAxis.multilevelLabels\$("#chart").ejChart({ primaryXAxis: { multilevelLabels: [{ visible: true }] } });	Property:primaryXAxis.multilevelLabels1let chart: Chart = new Chart({ primaryXAxis: { multilevelLabels: [{ }] } });chart.appendTo('#chart');
Default behaviour for multilevel Labels	Property:primaryXAxis.multilevelLabels\$("#chart").ejChart({ primaryXAxis: { multilevelLabels: [{ visible: true }] } });	Property:primaryXAxis.multilevelLabels1let chart: Chart = new Chart({ primaryXAxis: { multilevelLabels: [{ }] } });chart.appendTo('#chart');
Text alignment for multilevel Labels	Property:multiLevelLabels.textAlignment\$("#chart").ejChart({ primaryXAxis: { multilevelLabels: [{ textAlignment: 'Near' }] } });	Property:multilevelLabels.alignmentlet chart: Chart = new Chart({ primaryXAxis: { multilevelLabels: [{ alignment: 'Near' }] } });chart.appendTo('#chart');
Text overflow for multilevel Labels	Property:multiLevelLabels.textOverFlow\$("#chart").ejChart({ primaryXAxis: { multilevelLabels: [{ textOverFlow: 'Trim' }] } });	Property:multiLevelLabels.overFlowlet chart: Chart = new Chart({ primaryXAxis: { multilevelLabels: [{ overFlow: 'Trim' }] } });chart.appendTo('#chart');
Border for multilevel Labels	Property:multiLevelLabels.border\$("#chart").ejChart({ primaryXAxis: { multilevelLabels: [{ border: { width: 2, color: 'red', type: 'brace' } }] } });	Property:multiLevelLabels.borderlet chart: Chart = new Chart({ primaryXAxis: { multilevelLabels: [{ border: { width: 2, color: 'red', type: 'brace' } }] } });chart.appendTo('#chart');
Start value for label	Property:multiLevelLabels.start\$("#chart").ejChart({ primaryXAxis: { multilevelLabels: [{ start: 45 }] } });	Property:multiLevelLabels.categories.startlet chart: Chart = new Chart({ primaryXAxis: { multilevelLabels: [{ categories: [{ start: 45}] }] } });chart.appendTo('#chart');
End value for label	Property:multiLevelLabels.start\$("#chart").ejChart({ primaryXAxis: { multilevelLabels: [{ end: 45 }] } });	Property:multiLevelLabels.categories.endlet chart: Chart = new Chart({ primaryXAxis: { multilevelLabels: [{ categories: [{ end: 45}] }] } });chart.appendTo('#chart');
Text for label	Property:multiLevelLabels.text\$("#chart").ejChart({ primaryXAxis: { multilevelLabels: [{ text: 'Start' }] } });	Property:multiLevelLabels.categories.textlet chart: Chart = new Chart({ primaryXAxis: { multilevelLabels: [{ categories: [{ text: 'text' }] }] } });

		} }}}});chart.appendTo('#chart');
maximum text width for label	Property:multiLevelLabels.maximumTextWidth\$("<#chart").ejChart({ primaryXAxis: { multilevelLabels: [{ maximumTextWidth: 10 }] } }));	Property:multiLevelLabels.categories.maximumTextWidth let chart: Chart = new Chart({ primaryXAxis: { multilevelLabels: [{ categories: [{ maximumTextWidth: 20 }] }] } }));chart.appendTo('#chart');
level of labels	Property:multiLevelLabels.level\$("<#chart").ejChart({ primaryXAxis: { multilevelLabels: [{ level: 2 }] } }));	Not applicable. Categories are used

Methods

Behaviour	API in Essential JS 1	API in Essential JS 2
animation for series	Property:chart.animate\$("<#chart").ejChart({ animate: () { } });	Not applicable
Redraw for chart	Property:chart.redraw\$("<#chart").ejChart({ redraw: () { } });	Property:chart.refresh() let chart: Chart = new Chart({});chart.appendTo('#chart');chart.width = '400';chart.refresh();
Export	Property:chart.export\$("<#chart").ejChart({ export: () { } });	Property:chart.export() let chart: Chart = new Chart({});chart.export('JPEG', 'chart');chart.appendTo('#chart');
Print	Property:chart.print\$("<#chart").ejChart({ print: () { } });	Property:chart.print() let chart: Chart = new Chart({});chart.print('chart');chart.appendTo('#chart');
AddSeries	Not Applicable	Property:chart.addSeries() let chart: Chart = new Chart({});chart.appendTo('#chart');chart.addSeries();
RemoveSeries	Not Applicable	Property:chart.removeSeries() let chart: Chart = new Chart({});chart.appendTo('#chart');chart.removeSeries();

Events

Behaviour	API in Essential JS 1	API in Essential JS 2
Fires on annotation click	Property:annotationClick\$("<#chart").ejChart({ annotationClick: () { } });	Not applicable
Fires after animation	Property:animationComplete\$("<#chart").ejChart({ animationComplete: () { } });	Property:animationComplete() let chart: Chart = new Chart({

		animationComplete: () => { });chart.appendTo('#chart');
Fires on axis label click	Property:axisLabelClick\$("#chart").ejChart({ axisLabelClick: () { }});	Not applicable
Fires before axis label render	Property:axisLabelRendering\$("#chart").ejChart({ axisLabelRendering: () { }});	Property:axisLabelRender() let chart: Chart = new Chart({ axisLabelRender: () => { });chart.appendTo('#chart');
Fires on axis label mouseMove	Property:axisLabelMouseMove\$("#chart").ejChart({ axisLabelMouseMove: () { }});	Not applicable
Fires on axis label initialize	Property:axisLabelInitialize\$("#chart").ejChart({ axisLabelInitialize: () { }});	Not applicable
Fires before axis range calculation	Property:axesRangeCalculate\$("#chart").ejChart({ axesRangeCalculate: () { }});	Property:axisRangeCalculated() let chart: Chart = new Chart({ axisRangeCalculated: () => { });chart.appendTo('#chart');
Fires on axis title rendering	Property:axisTitleRendering\$("#chart").ejChart({ axisTitleRendering: () { }});	Not applicable
Fires on after chart resize	Property:afterResize\$("#chart").ejChart({ afterResize: () { }});	Not applicable
Fires on before chart resize	Property:beforeResize\$("#chart").ejChart({ beforeResize: () { }});	Property:resized let chart: Chart = new Chart({ resized: () => { });chart.appendTo('#chart');
Fires on chart click	Property:chartClick\$("#chart").ejChart({ chartClick: () { }});	Property:chartMouseClicked let chart: Chart = new Chart({ chartMouseClicked: () => { });chart.appendTo('#chart');

Fires on chart mouse move	Property:chartMouseMove\$("#chart").ejChart({ chartMouseMove: () { }});	Property:chartMouseMovelet chart: Chart = new Chart({ chartMouseMove: () => { }});chart.appendTo('#chart');
Fires on chart mouse leave	Property:chartMouseLeave\$("#chart").ejChart({ chartMouseLeave: () { }});	Property:chartMouseLeavelet chart: Chart = new Chart({ chartMouseLeave: () => { }});chart.appendTo('#chart');
Fires on before chart double click	Property:chartDoubleClick\$("#chart").ejChart({ chartDoubleClick: () { }});	Not applicable
Fires on chart mouse up	Not Applicable	Property:chartmouseUplet chart: Chart = new Chart({ chartmouseUp: () => { }});chart.appendTo('#chart');
Fires on chart mouse down	Not Applicable	Property:chartmouseDownlet chart: Chart = new Chart({ chartmouseDown: () => { }});chart.appendTo('#chart');
Fires during the calculation of chart area bounds. You can use this event to customize the bounds of chart area	Property:chartAreaBoundsCalculate\$("#chart").ejChart({ chartAreaBoundsCalculate: () { }});	Not applicable
Fires when the dragging is started	Property:dragStart\$("#chart").ejChart({ dragStart: () { }});	Not applicable

Fires while dragging	Property:dragging\$("#chart").ejChart({ dragging: () { }});	Not applicable
Fires when the dragging is completed	Property:dragEnd\$("#chart").ejChart({ dragEnd: () { }});	Property:dragCompletelet chart: Chart = new Chart({ dragComplete: () => { }});chart.appendTo('#char t');
Fires when chart is destroyed completely.	Property:destroy\$("#chart").ejChart({ destroy: () { }});	Not applicable
Fires after chart is created.	Property:create\$("#chart").ejChart({ create: () { }});	Property:loadedlet chart: Chart = new Chart({ loaded: () => { }});chart.appendTo('#char t');
Fires before rendering the data labels.	Property:displayTextRendering\$("#chart").ejChart ({ displayTextRendering: () { }});	Property:textRenderlet chart: Chart = new Chart({ textRender: () => { }});chart.appendTo('#char t');
Fires, when error bar is rendering.	Property:errorBarRendering\$("#chart").ejChart({ errorBarRendering: () { }});	Not applicable
Fires during the calculation of legend bounds.	Property:legendBoundsCalculate\$("#chart").ejChar t({ legendBoundsCalculate: () { }});	Not applicable
Fires on clicking the legend item.	Property:legendItemClick\$("#chart").ejChart({ legendItemClick: () { }});	Not applicable
Fires when moving mouse over legend item	Property:legendItemMouseMove\$("#chart").ejCha rt({ legendItemMouseMove: () { }});	Not applicable

Fires before rendering the legend item.	Property:legendItemRendering\$("#chart").ejChart({ legendItemRendering: () { }});	Property:legendRenderlet chart: Chart = new Chart({ legendRender: () => { }});chart.appendTo('#char t');
Fires before loading the chart.	Property:load\$("#chart").ejChart({ load: () { }});	Property:loadlet chart: Chart = new Chart({ load: () => { }});chart.appendTo('#char t');
Fires, when multi level labels are rendering.	Property:multiLevelLabelRendering\$("#chart").ejChart({ multiLevelLabelRendering: () { }});	Property:axisMultiLabelRender let chart: Chart = new Chart({ axisMultiLabelRender : () => { }});chart.appendTo('#char t');
Fires on clicking a point in chart.	Property:pointRegionClick\$("#chart").ejChart({ pointRegionClick: () { }});	Property:pointClick let chart: Chart = new Chart({ pointClick : () => { }});chart.appendTo('#char t');
Fires when mouse is moved over a point.	Property:pointRegionMouseMove\$("#chart").ejChart({ pointRegionMouseMove: () { }});	Property:pointMove let chart: Chart = new Chart({ pointMove : () => { }});chart.appendTo('#char t');
Fires before rendering chart.	Property:preRender\$("#chart").ejChart({ preRender: () { }});	Not applicable
Fires when point render.	Not Applicable	Property:pointRender let chart: Chart = new Chart({ pointRender : () => { }});chart.appendTo('#char t');
Fires after selected the data in chart.	Property:rangeSelected\$("#chart").ejChart({ rangeSelected: () { }});	Not applicable
Fires after selecting a series.	Property:seriesRegionClick\$("#chart").ejChart({ seriesRegionClick: () { }});	Not applicable

Fires before rendering a series.	Property:seriesRendering\$("#chart").ejChart({ seriesRendering: () { }});	Property:seriesRender let chart: Chart = new Chart({ seriesRender : () => { }});chart.appendTo('#chart');
Fires before rendering the marker symbols.	Property:symbolRendering\$("#chart").ejChart({ symbolRendering: () { }});	Not applicable
Fires before rendering the trendline	Property:trendlineRendering\$("#chart").ejChart({ trendlineRendering: () { }});	Not applicable
Fires before rendering the Chart title.	Property:titleRendering\$("#chart").ejChart({ titleRendering: () { }});	Not applicable
Fires before rendering the Chart sub title.	Property:subTitleRendering\$("#chart").ejChart({ subTitleRendering: () { }});	Not applicable
Fires before rendering the tooltip.	Property:toolTipInitialize\$("#chart").ejChart({ toolTipInitialize: () { }});	Property:tooltipRender let chart: Chart = new Chart({ tooltipRender : () => { }});chart.appendTo('#chart');
Fires before rendering crosshair tooltip in axis	Property:trackAxisToolTip\$("#chart").ejChart({ trackAxisToolTip: () { }});	Not applicable
Fires before rendering trackball tooltip.	Property:trackToolTip\$("#chart").ejChart({ trackToolTip: () { }});	Not applicable

Event triggered when scroll starts.	Property:scrollStart\$("#chart").ejChart({ scrollStart: () { }});	Property:scrollStart let chart: Chart = new Chart({ scrollStart : () => { }});chart.appendTo('#chart');
Event triggered when scroll ends.	Property:scrollEnd\$("#chart").ejChart({ scrollEnd: () { }});	Property:scrollEndlet chart: Chart = new Chart({ scrollEnd: () => { }});chart.appendTo('#chart');
Event triggered when scroll changes.	Property:scrollChange\$("#chart").ejChart({ scrollChange: () { }});	Property:scrollChangelet chart: Chart = new Chart({ scrollChange: () => { }});chart.appendTo('#chart');
Fires while performing rectangle zooming in chart.	Property:zoomComplete\$("#chart").ejChart({ zoomComplete: () { }});	Property:zoomCompletele t chart: Chart = new Chart({ zoomComplete: () => { }});chart.appendTo('#chart'); ## Chart properties
Behaviour	API in Essential JS 1	API in Essential JS 2
selected data index	Property:selectedDataPointIndexes\$("#chart").ejChart({ selectedDataPointIndexes: [{ seriesIndex: 0, pointIndex: 1}]});	Property:selectedDataIndexeslet chart: Chart = new Chart({selectedDataIndexes: [{ series: 0, point: 1}]});chart.appendTo('#chart');
sideBySideSeries Placement for column based series	Property:sideBySideSeriesPlacement\$("#chart").ejChart({ sideBySideSeriesPlacement});	Property:sideBySidePlacementlet chart: Chart = new Chart({ sideBySidePlacement: true});chart.appendTo('#chart');
ZoomSettings	Property:zooming\$("#chart").ejChart({ zooming: { enable: true, enableDeferredZoom: true, enablePinch: true, enableMouseWheel: true, enableScrollBar: true, toolbarItems: [], type: 'X' }});	Property:zoomSettingslet chart: Chart = new Chart({ zoomSettings: { enable: true, enablePinchZooming: true, enableDeferredZooming: true, enableMouseWheelZooming: true, enableSelectionZooming: true, enableScrollBar: true }});chart.appendTo('#chart');
Background color of the chart	Property:background\$("#container").ejChart({ background: 'transparent'});	Property:backgroundlet chart: Chart = new Chart({ background: '#EEFFCC'});chart.appendTo('#chart');

URL of the image to be used as chart background.	Property:backGroundImageUrl \$("#container").ejChart({ backGroundImageUrl : '../images/chart/wheat.png'}) ;	Not Applicable
Customizing border of the chart	Property:border \$("#container").ejChart({ border: { width: 2, color: '#CCEEFF', opacity: 0.5}});	Property:borderlet chart: Chart = new Chart({ border: { width: 2, color: '#CCEEFF'}});chart.appendTo('#char t');
This provides options for customizing export settings	Property:exportSettings\$("#containe r").ejChart({ exportSettings: { filename : "chart", angle: '45' }});	Property:export()let chart: Chart = new Chart({ border: { width: 2, color: '#CCEEFF'}});chart.appendTo('#char t');chart.export(type, fileName);
ChartArea customization	Property:chartArea\$("#container") .ejChart({ chartArea: { background: 'transparent', border: { opacity: 0.3, color: 'red', width: 2}}});	Property:chartArealet chart: Chart = new Chart({ chartArea: { background: 'transparent', border: { width: 2, color: '#CCEEFF' }});chart.appendTo('#chart');chart .export(type, fileName);

Rows

<!-- markdownlint-disable MD033 -->

Behaviour	API in Essential JS 1	API in Essential JS 2
rows in chart	Property:rowDefinitions\$("#chart").ejC hart({ rowDefinitions: []});	Property:rowslet chart: Chart = new Chart({ rows: []});chart.appendTo('#chart');
unit	Property:unit \$("#container").ejChart({ rowDefinitions :[{unit : "percentage"}]});	Not Applicable
height of rows in chart	Property:rowHeight\$("#chart").ejChart ({ rowDefinitions: [{ rowHeight: '50%'}]});	Property:heightlet chart: Chart = new Chart({ rows: [{ height: '300'}]});chart.appendTo('#chart
Line customization	Property:lineColor, lineWidth\$("#chart").ejChart({ rowDefinitions: [{ rowHeight: '50%', lineColor: 'brown', lineWidth: 2}]});	Property:borderlet chart: Chart = new Chart({ rows: [{ height: '300', border: { width: 2, color: 'brown'}}]});chart.appendTo('#ch art');

Series

<!-- markdownlint-disable MD033 -->

Behaviour	API in Essential JS 1	API in Essential JS 2
bearFillColor	Property:bearFillColor\$("#chart").ejChart({ series: [{bearFillColor: 'red' }]});	Property:bearFillColorlet chart: Chart = new Chart({ series: [{bearFillColor: 'red' }]});chart.appendTo('#chart');
Border	Property:border\$("#chart").ejChart({ series: [{ border: { color: 'red', width: 2, dashArray: '10, 5' } }]});	Property:rowslet chart: Chart = new Chart({ series: [{ border: { color: 'red', width: 2} }]});chart.appendTo('#chart');
BoxPlotMode	Property:boxPlotMode\$("#chart").ejChart({ series: [{ boxPlotMode: 'inclusive' }]});	Property:rowslet chart: Chart = new Chart({ series: [{ boxPlotMode: 'Inclusive' }]});chart.appendTo('#chart');
Minimum radius of Bubble series	Property:bubbleOptions.minRadius\$("#chart").ejChart({ series: [{ bubbleOptions: { minRadius: 2 } }]});	Property:minRadiuslet chart: Chart = new Chart({ series: [{ minRadius: 2 }]});chart.appendTo('#chart');
Maximum radius of Bubble series	Property:bubbleOptions.maxRadius\$("#chart").ejChart({ series: [{ bubbleOptions: { maxRadius: 10 } }]});	Property:maxRadiuslet chart: Chart = new Chart({ series: [{ maxRadius: 2 }]});chart.appendTo('#chart');
bullFillColor	Property:bullFillColor\$("#chart").ejChart({ series: [{bullFillColor: 'red' }]});	Property:bullFillColorlet chart: Chart = new Chart({ series: [{bullFillColor: 'red' }]});chart.appendTo('#chart');
Cardinal spline tension for spline series	Property:cardinalSplineTension\$("#chart").ejChart({ series: [{ cardinalSplineTension: 0.5 }]});	Property:cardinalSplineTensionlet chart: Chart = new Chart({ series: [{ cardinalSplineTension: 0.5 }]});chart.appendTo('#chart');
Column Width for rectangle series	Property:columnWidth\$("#chart").ejChart({ series: [{ columnWidth: 0.5 }]});	Property:columnWidthlet chart: Chart = new Chart({ series: [{ columnWidth: 0.5 }]});chart.appendTo('#chart');

Column spacing for rectangle series	Property:columnSpacing\$("#chart").ejChart({ series: [{ columnSpacing: 0.5 }]});	Property:columnSpacinglet chart: Chart = new Chart({ series: [{ columnSpacing: 0.5 }]});chart.appendTo('#chart');
Topleft radius for rectangle series	Property:cornerRadius.topLeft\$("#chart").ejChart({ series: [{ topLeft: 0 }]});	Property:cornerRadius.topLeftlet chart: Chart = new Chart({ series: [{ topLeft: 0 }]});chart.appendTo('#chart');
topRight radius for rectangle series	Property:cornerRadius.topRight\$("#chart").ejChart({ series: [{ topRight: 0 }]});	Property:cornerRadius.topRightlet chart: Chart = new Chart({ series: [{ topRight: 0 }]});chart.appendTo('#chart');
bottomRight radius for rectangle series	Property:cornerRadius.bottomRight\$("#chart").ejChart({ series: [{ bottomRight: 0 }]});	Property:cornerRadius.bottomRightlet chart: Chart = new Chart({ series: [{ bottomRight: 0 }]});chart.appendTo('#chart');
bottomLeft radius for rectangle series	Property:cornerRadius.bottomLeft\$("#chart").ejChart({ series: [{ bottomLeft: 0 }]});	Property:cornerRadius.bottomLeftlet chart: Chart = new Chart({ series: [{ bottomLeft: 0 }]});chart.appendTo('#chart');
DashArray property	Property:dashArray\$("#chart").ejChart({ series: [{ dashArray: '10, 5' }]});	Property:dashArraylet chart: Chart = new Chart({ series: [{ dashArray: '10, 5' }]});chart.appendTo('#chart');
DataSource for series	Property:dataSource\$("#chart").ejChart({ series: [{ dataSource: [] }]});	Property:dashArraylet chart: Chart = new Chart({ series: [{ dataSource: [] }]});chart.appendTo('#chart');
Draw type for Polar series	Property:drawType\$("#chart").ejChart({ series: [{ drawType: 'Line' }]});	Property:drawTypelet chart: Chart = new Chart({ series: [{ drawType: 'Line' }]});chart.appendTo('#chart');

EmptyPointSettings for series	Property:emptyPointSettings.visible\$("#chart").ejChart({ series: [{ emptyPointSettings: { visible: false } }]});	Not Applicable
Empty Point Display mode	Property:emptyPointSettings.displayMode\$("#chart").ejChart({ series: [{ displayMode: 'gap' }]});	Property:emptyPointSettings.displayModelet chart: Chart = new Chart({ series: [{ displayMode: 'Average' }]});chart.appendTo('#chart');
Empty Point color	Property:emptyPointSettings.color\$("#chart").ejChart({ series: [{ color: 'red' }]});	Property:emptyPointSettings.filllet chart: Chart = new Chart({ series: [{ fill: 'red' }]});chart.appendTo('#chart');
Empty Point Border	Property:emptyPointSettings.border\$("#chart").ejChart({ series: [{ emptyPointSettings: { color: 'red', width: 2 } }]});	Property:filllet chart: Chart = new Chart({ series: [{ emptyPointSettings: { color: 'red', width: 2 } }]});chart.appendTo('#chart');
Enable animation for series	Property:enableAnimation\$("#chart").ejChart({ series: [{ enableAnimation: true }]});	Property:animation.enablelet chart: Chart = new Chart({ series: [animation: { enable: false }]});chart.appendTo('#chart');
Animation duration for series	Property:animationDuration\$("#chart").ejChart({ series: [{ animationDuration: 1000 }]});	Property:animation.durationlet chart: Chart = new Chart({ series: [animation: { duration: 1000 }]});chart.appendTo('#chart');
Animation delay for series	Not Applicable	Property:animation.durationlet chart: Chart = new Chart({ series: [animation: { delay: 100 }]});chart.appendTo('#chart');
Drag settings for series	Property:dragSettings\$("#chart").ejChart({ series: [{ dragSettings: { mode: 'X' } }]});	Not Applicable
Errorbar settings for series	Property:errorBarSettings\$("#chart").ejChart({ series: [{ errorBarSettings: { } }]});	Property:errorBarSettingslet chart: Chart = new Chart({ series: [{errorBarSettings: { } }]});

Closed series	Property:isClosed\$("#chart").ejChart({ series: [{ isClosed: true }]});	Property:isClosedlet chart: Chart = new Chart({ series: [{ isClosed: true }]});
Stacking Property for series	Property:isStacking\$("#chart").ejChart({ series: [{ isStacking: true }]});	Not Applicable
Line cap for series	Property:lineCap\$("#chart").ejChart({ series: [{ lineCap: 'butt' }]});	Not Applicable
Line join for series	Property:lineJoin\$("#chart").ejChart({ series: [{ lineJoin: 'round' }]});	Not Applicable
Opacity for series	Property:opacity\$("#chart").ejChart({ series: [{ opacity: 0.7 }]});	Property:errorBarSettingslet chart: Chart = new Chart({ series: [{ opacity: 0.7 }]});
Outlier settings of series	Property:outLierSettings\$("#chart").ejChart({ series: [{ outLierSettings: { shape: 'rectangle' , size: { height: 30, width: 20} }]});	Not Applicable
Palette	Property:palette\$("#chart").ejChart({ series: [{ palette: "ColorFieldName" }]});	Property:pointColorMappinglet chart: Chart = new Chart({ series: [{ pointColorMapping: 'color' }]});chart.appendTo('#chart');
Positive fill for waterfall series	Property:positiveFill\$("#chart").ejChart({ series: [{ positiveFill: "red" }]});	Property:pointColorMappinglet chart: Chart = new Chart({ series: [{ pointColorMapping: 'color' }]});chart.appendTo('#chart');
Show average value in box and whisker series	Property:showMedian\$("#chart").ejChart({ series: [{ showMedian: true }]});	Property:pointColorMappinglet chart: Chart = new Chart({ series: [{ showMean: false }]});chart.appendTo('#chart');
To group the series of stacking collection.	Property:stackingGroup\$("#chart").ejChart({ series: [{ stackingGroup: 'group' }]});	Property:stackingGrouplet chart: Chart = new Chart({ series: [{ stackingGroup: 'group' }]});chart.appendTo('#chart');
Specifies the type of the series to	Property:type\$("#chart").ejChart({ series: [{ type: 'Line' }]});	Property:typetlet chart: Chart = new Chart({ series: [{ type: 'Line' }

render in chart.		<code>});});chart.appendTo('#chart');</code>
Defines the visibility of the series.	Property:visibility <code>\$("#chart").ejChart({ series: [{ visibility: true }];});</code>	Property:visible <code>let chart: Chart = new Chart({ series: [{ visible: true }];});chart.appendTo('#chart');</code>
Enables or disables the visibility of legend item.	Property:visibleOnLegend <code>\$("#chart").ejChart({ series: [{ visibleOnLegend : true }];});</code>	Property:toggleVisibility <code>let chart: Chart = new Chart({ legendSettings: [{ toggleVisibility: true }];});chart.appendTo('#chart');</code>
Specifies the different types of spline curve.	Property:splineType <code>\$("#chart").ejChart({ series: [{ splineType : 'Natural' }];});</code>	Property:splineType <code>let chart: Chart = new Chart({ legendSettings: [{ splineType: 'Natural' }];});chart.appendTo('#chart');</code>
Specifies the name of the x-axis that has to be associated with this series. Add an axis instance with this name to axes collection.	Property:xAxisName <code>\$("#chart").ejChart({ series: [{ xAxisName : 'secondaryXAxis' }];});</code>	Property:xAxisName <code>let chart: Chart = new Chart({ series: [{ xAxisName: 'secondaryXAxis' }];});chart.appendTo('#chart');</code>
Name of the property in the datasource that contains x value for the series.	Property:xName <code>\$("#chart").ejChart({ series: [{ xName : 'x' }];});</code>	Property:xName <code>let chart: Chart = new Chart({ series: [{ xName: 'x' }];});chart.appendTo('#chart');</code>
Specifies the name of the y-axis that has to be associated with this series. Add an axis instance with this	Property:yAxisName <code>\$("#chart").ejChart({ series: [{ yAxisName : 'secondaryYAxis' }];});</code>	Property:yAxisName <code>let chart: Chart = new Chart({ series: [{ yAxisName: 'secondaryYAxis' }];});chart.appendTo('#chart');</code>

name to axes collection.		
Name of the property in the datasource that contains y value for the series.	Property:yName \$("#chart").ejChart({ series: [{ yName : 'y' }]});	Property:yName let chart: Chart = new Chart({ series: [{ yName: 'y' }]});chart.appendTo('#chart');
Name of the property in the datasource that contains high value for the series.	Property:high \$("#chart").ejChart({ series: [{ high : 'y' }]});	Property:high let chart: Chart = new Chart({ series: [{ high: 'y' }]});chart.appendTo('#chart');
Name of the property in the datasource that contains low value for the series.	Property:low \$("#chart").ejChart({ series: [{ low : 'y' }]});	Property:low let chart: Chart = new Chart({ series: [{ low: 'y' }]});chart.appendTo('#chart');
Name of the property in the datasource that contains close value for the series.	Property:close \$("#chart").ejChart({ series: [{ close : 'y' }]});	Property:close let chart: Chart = new Chart({ series: [{ close: 'y' }]});chart.appendTo('#chart');
Name of the property in the datasource that contains open value for the series.	Property:open \$("#chart").ejChart({ series: [{ open : 'y' }]});	Property:open let chart: Chart = new Chart({ series: [{ open: 'y' }]});chart.appendTo('#chart');
Option to add trendlines to chart.	Property:trendLines \$("#chart").ejChart({ series: [{ trendLines : [{}]}]});	Property:trendLines let chart: Chart = new Chart({ series: [{ trendLines : [{}]}]});chart.appendTo('#chart');

Options for customizing the appearance of the series or data point while highlighting.	Property:highlightSettings\$("#chart").ejChart({ series: [{ highlightSettings : {} }]});	Not applicable.
Options for customizing the appearance of the series/data point on selection.	Property:selectionSettings \$("#chart").ejChart({ series: [{ selectionSettings : {} }]});	Not applicable.

marker

<!-- markdownlint-disable MD033 -->

visibility of marker	Property:visible\$("#chart").ejChart({ series: [{ marker: { visible: true } }]});	Property:visiblelet chart: Chart = new Chart({ series: [{ marker: { visible: false } }];});chart.appendTo('#c hart');
Fill for marker	Property:fill\$("#chart").ejChart({ series: [{ marker: { fill : 'red' } }]});	Property:visiblelet chart: Chart = new Chart({ series: [{ marker: { fill : 'red' } }];});chart.appendTo('#c hart');
Opacity for marker	Property:opacity\$("#chart").ejChart({ series: [{ marker: { opacity : 0.5 } }]});	Property:opacitylet chart: Chart = new Chart({ series: [{ marker: { opacity : 0.5 } }];});chart.appendTo('#c hart');
Shape of marker	Property:shape\$("#chart").ejChart({ series: [{ marker: { shape : 'Circle' } }]});	Property:shapelet chart: Chart = new Chart({ series: [{ marker: { shape : 'Triangle' } }];});chart.appendTo('#c hart');
ImageUrl of marker	Property:imageUrl\$("#chart").ejChart({ series: [{ marker: { imageUrl : '' } }]});	Property:imageUrllet chart: Chart = new Chart({ series: [{ marker: {

		<pre>imageUrl : '' } });});chart.appendTo('#c hart);</pre>
Border of marker	<pre>Property:border\$("#chart").ejChart({ series: [{ marker: { border : { width: 2, color: 'red' } } }]});});</pre>	<pre>Property:shapelet chart: Chart = new Chart({ series: [{ marker: { border : { width: 2, color: 'red' } } }]});});chart.appendTo('#c hart);</pre>
Height of marker	<pre>Property:size.height\$("#chart").ejChart({ series: [{ marker: { size: { height: 30 } } }]});});</pre>	<pre>Property:heightlet chart: Chart = new Chart({ series: [{ marker: { height: 25 } }]});});chart.appendTo('#c hart);</pre>
Width of marker	<pre>Property:size.width\$("#chart").ejChart({ series: [{ marker: { size: { width: 30 } } }]});});</pre>	<pre>Property:widthlet chart: Chart = new Chart({ series: [{ marker: { width: 25 } }]});});chart.appendTo('#c hart);</pre>
Width of marker	<pre>Property:size.width\$("#chart").ejChart({ series: [{ marker: { size: { width: 30 } } }]});});</pre>	<pre>Property:widthlet chart: Chart = new Chart({ series: [{ marker: { width: 25 } }]});});chart.appendTo('#c hart);</pre>
DataLabelSettings of marker	<pre>Property:marker.dataLabel\$("#chart").ejChart({ series: [{ marker: { dataLabel: { } } }]});});</pre>	<pre>Property:marker.dataLabellet chart: Chart = new Chart({ series: [{ marker: { dataLabel: { } } }]});});chart.appendTo('#c hart);</pre>
Visibility of dataLabel	<pre>Property:dataLabel.visible\$("#chart").ejChart({ series: [{ marker: { dataLabel: { visible: true } } }]});});</pre>	<pre>Property:dataLabel.visiblelet chart: Chart = new Chart({ series: [{ marker: { dataLabel: { visible: true } } }]});});chart.appendTo('#c hart);</pre>
Text mapping name of dataLabel	<pre>Property:dataLabel.textMappingName\$("#chart").ejChart({ series: [{ marker: { dataLabel: { textMappingName: '' } } }]});});</pre>	<pre>Property:dataLabel.namelet chart: Chart = new Chart({ series: [{ marker: { dataLabel: { name: '' } } }]});});chart.appendTo('#c hart);</pre>

Fill color of data label	<code>Property:dataLabel.fill\$("#chart").ejChart({ series: [{ marker: {dataLabel: { fill: 'pink' } } }]});</code>	<code>Property:dataLabel.filllet chart: Chart = new Chart({ series: [{ marker: { dataLabel: { fill: 'pink' } } }]});chart.appendTo('#c hart');</code>
Opacity of data label	<code>Property:dataLabel.opacity\$("#chart").ejChart({ series: [{ marker: {dataLabel: { opacity: 0.6 } } }]});</code>	<code>Property:dataLabel.filllet chart: Chart = new Chart({ series: [{ marker: { dataLabel: { opacity: 0.4 } } }]});chart.appendTo('#c hart');</code>
Text position of data label	<code>Property:dataLabel.textPosition\$("#chart").ejChar t({ series: [{ marker: {dataLabel: { textPosition: 'middle' } } }]});</code>	<code>Property:dataLabel.positionlet chart: Chart = new Chart({ series: [{ marker: { dataLabel: { position: 'Top' } } }]});chart.appendTo('#c hart');</code>
Alignment of data label	<code>Property:dataLabel.verticalAlignment\$("#chart").ej Chart({ series: [{ marker: {dataLabel: { verticalAlignment: 'near' } } }]});</code>	<code>Property:dataLabel.alignmentle t chart: Chart = new Chart({ series: [{ marker: { dataLabel: { alignment: 'Near' } } }]});chart.appendTo('#c hart');</code>
Border of data label	<code>Property:dataLabel.border\$("#chart").ejChart({ series: [{ marker: {dataLabel: { border: { color: 'blue', width: 2, opacity: 0.4 } } } }]});</code>	<code>Property:dataLabel.alignmentle t chart: Chart = new Chart({ series: [{ marker: { dataLabel: { border: { color: 'blue', width: 2 } } } }]});chart.appendTo('#c hart');</code>
Offset for data label	<code>Property:dataLabel.offset\$("#chart").ejChart({ series: [{ marker: {dataLabel: { offset: { x: 5, y: 6 } } } }]});</code>	Not Applicable
Margin of data label	<code>Property:dataLabel.border\$("#chart").ejChart({ series: [{ marker: {dataLabel: { margin: { top: 10, bottom: 10, left: 10, right: 10 } } } }]});</code>	<code>Property:dataLabel.marginlet chart: Chart = new Chart({ series: [{ marker: { dataLabel: { margin: { top: 10, bottom: 10, left: 10, right: 10 } } } }]});chart.appendTo('#c hart');</code>

Font of data label	Property:dataLabel.border\$("#chart").ejChart({ series: [{ marker: {dataLabel: { font: { fontFamily: 'SegoeUI', fontStyle: 'italic', fontWeight: '600', opacity: 0.5, size: 12, color: 'red' }} } }]});	Property:dataLabel.marginlet chart: Chart = new Chart({ series: [{ marker: {dataLabel: { font: { fontFamily: 'SegoeUI', fontStyle: 'italic', fontWeight: '600', opacity: 0.5, size: 12, color: 'red' }} } }]});chart.appendTo('#c hart');
HTML template in dataLabel	Property:dataLabel.template\$("#chart").ejChart({ series: [{ marker: {dataLabel: { template: 'Chart' } } }]});	Property:dataLabel.templatelet t chart: Chart = new Chart({ series: [{ marker: {dataLabel: { template: 'Chart' } } } }]});chart.appendTo('#c hart');
Rounded corner radius X	Not Applicable	Property:dataLabel.rxlet chart: Chart = new Chart({ series: [{ marker: {dataLabel: { rx: 10 } } }]});chart.appendTo('#c hart');
Rounded corner radius Y	Not Applicable	Property:dataLabel.rylet chart: Chart = new Chart({ series: [{ marker: {dataLabel: { ry: 10 } } }]});chart.appendTo('#c hart');
Maximum Label width for data label	Property:dataLabel.maximumLabelWidth\$("#chart").ejChart({ series: [{ marker: {dataLabel: { maximumLabelWidth: 20 } } }]});	Not Applicable
Enable wrapping of text for data label	Property:dataLabel.enableWrap\$("#chart").ejChart({ series: [{ marker: {dataLabel: { enableWrap: true } } }]});	Not Applicable
To show contrast color for data label	Property:dataLabel.showContrastColor\$("#chart").ejChart({ series: [{ marker: {dataLabel: { showContrastColor: true } } }]});	Not Applicable

To show edge label for data label	Property: <code>dataLabel.showEdgeLabels\$("#chart").ejChart({ series: [{ marker: {dataLabel: { showEdgeLabels: true }} }]});</code>	Not Applicable
-----------------------------------	---	----------------

TrendLines

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Trendlines settings	Property: <code>series.trendLines\$("#chart").ejChart({ series: [{ trendLines: [] }]});</code>	Property: <code>series.trendLines</code> let chart: Chart = new Chart({ series: [{ trendLines: [] }]}); chart.appendTo('#chart');
Visibility of trendline	Property: <code>trendLines.visibility\$("#chart").ejChart({ series: [{ trendLines: [visibility: true] }]});</code>	Not applicable
Type of trendLine	Property: <code>trendLines.type\$("#chart").ejChart({ series: [{ trendLines: [type: 'linear'] }]});</code>	Property: <code>trendLines.type</code> let chart: Chart = new Chart({ series: [{ trendLines: [type: 'Polynomial'] }]}); chart.appendTo('#chart');
Name of trendLine	Property: <code>trendLines.name\$("#chart").ejChart({ series: [{ trendLines: [name: 'trendLine'] }]});</code>	Property: <code>trendLines.name</code> let chart: Chart = new Chart({ series: [{ trendLines: [name: 'trendLine'] }]}); chart.appendTo('#chart');
Period of trendLine	Property: <code>trendLines.period\$("#chart").ejChart({ series: [{ trendLines: [period: 45] }]});</code>	Property: <code>trendLines.period</code> let chart: Chart = new Chart({ series: [{ trendLines: [period: 45] }]}); chart.appendTo('#chart');
Polynomial order for Polynomial type trendLines	Property: <code>trendLines.polynomialOrder\$("#chart").ejChart({ series: [{ trendLines: [polynomialOrder: 3] }]});</code>	Property: <code>trendLines.polynomialOrder</code> let chart: Chart = new Chart({ series: [{ trendLines: [polynomialOrder: 3] }]}); chart.appendTo('#chart');
Backward forecast for	Property: <code>trendLines.backwardforecast\$("#chart").ejChart({ series: [{ trendLines: [backwardforecast: 3] }]});</code>	Property: <code>trendLines.backwardforecast</code> let chart: Chart = new Chart({ series: [{ trendLines: [

trendLines		backwardforecast: 3]]]]);chart.appendTo('#chart');
Forward forecast for trendLines	Property:trendLines.forwardForecast\$("#chart").ejChart({ series: [{ trendLines: [forwardForecast: 3]}]]]);	Property:trendLines.forwardForecastlet chart: Chart = new Chart({ series: [{ trendLines: [forwardForecast: 3]}]]]);chart.appendTo('#chart');
Fill for trendLines	Property:trendLines.fill\$("#chart").ejChart({ series: [{ trendLines: [fill: '#EEFFCC']}]]]);	Property:trendLines.filllet chart: Chart = new Chart({ series: [{ trendLines: [fill: 'EEFFCC']}]]]);chart.appendTo('#chart');
Width for trendLines	Property:trendLines.width\$("#chart").ejChart({ series: [{ trendLines: [width: 2]}]]]);	Property:trendLines.widthlet chart: Chart = new Chart({ series: [{ trendLines: [width: 2]}]]]);chart.appendTo('#chart');
Intercept value for trendLines	Property:trendLines.intercept\$("#chart").ejChart({ series: [{ trendLines: [intercept: 2]}]]]);	Property:trendLines.interceptlet chart: Chart = new Chart({ series: [{ trendLines: [intercept: 2]}]]]);chart.appendTo('#chart');
Legend shape for trendLines	Not Applicable	Property:trendLines.legendShapelet chart: Chart = new Chart({ series: [{ trendLines: [legendShape: 'Rectangle']}]]]);chart.appendTo('#chart');
Animation settings for trendLines	Not Applicable	Property:trendLines.animationlet chart: Chart = new Chart({ series: [{ trendLines: [animation: { enable: true }]}]]]);chart.appendTo('#chart');
Marker settings for trendLines	Not Applicable	Property:trendLines.markerlet chart: Chart = new Chart({ series: [{ trendLines: [{marker: { visible: true }]}]]]);chart.appendTo('#chart');
Tooltip for trendLines	Property:trendLines.tooltip\$("#chart").ejChart({ series: [{ trendLines: [{ tooltip: { } }]}]]]);	Property:trendLines.enableTooltiplet chart: Chart = new Chart({ series: [{ trendLines: [{enableTooltip: true }]}]]]);chart.appendTo('#chart');

DashArray for trendLines	Property:trendLines.dashArray\$("#chart").ejChart({ series: [{ trendLines: [{ dashArray: '10, 5' }]}]});	Not Applicable.
Visible on legend for trendLines	Property:trendLines.visibleOnLegend\$("#chart").ejChart({ series: [{ trendLines: [{ visibleOnLegend: true }]}]});	Not Applicable.

StripLines

<!-- markdownlint-disable MD033 -->

Behaviour	API in Essential JS 1	API in Essential JS 2
Default behaviour for striplines	Property:primaryXAxis.stripLines\$("#chart").ejChart({ primaryXAxis: { stripLines: [{ visible: true }]}]});	Property:primaryXAxis.stripLineslet chart: Chart = new Chart({ primaryXAxis: { stripLines: [{ visible: true }]}]};chart.appendTo('#chart');
border for stripline	Property:stripLines.borderColor\$("#chart").ejChart({ primaryXAxis: { stripLines: [{ borderColor: 'pink' }]}]});	Property:stripLines.borderlet chart: Chart = new Chart({ primaryXAxis: { stripLines: [{ border: { color: 'red', width: 2 } }]}]};chart.appendTo('#chart');
Background color for stripline	Property:stripLines.color\$("#chart").ejChart({ primaryXAxis: { stripLines: [{ color: 'pink' }]}]});	Property:stripLines.borderlet chart: Chart = new Chart({ primaryXAxis: { stripLines: [{ color: 'red' }]}]};chart.appendTo('#chart');
Start value for stripline	Property:stripLines.start\$("#chart").ejChart({ primaryXAxis: { stripLines: [{ start: 10 }]}]});	Property:stripLines.startlet chart: Chart = new Chart({ primaryXAxis: { stripLines: [{ start: 5 }]}]};chart.appendTo('#chart');
End value for stripline	Property:stripLines.end\$("#chart").ejChart({ primaryXAxis: { stripLines: [{ end: 10 }]}]});	Property:stripLines.endlet chart: Chart = new Chart({ primaryXAxis: { stripLines: [{ end: 5 }]}]};chart.appendTo('#chart');
Startfrom Axis for stripline	Property:stripLines.startFromAxis\$("#chart").ejChart({ primaryXAxis: { stripLines: [{ startFromAxis: true }]}]});	Property:stripLines.startFromAxislet chart: Chart = new Chart({ primaryXAxis: { stripLines: [{ startFromAxis: true }]}]};

		<code>true}}}});chart.appendTo('#chart');</code>
Text in stripline	<code>Property:stripLines.text\$("#chart").ejChart({ primaryXAxis: { stripLines: [{ text: 'StripLine; }]}));</code>	<code>Property:stripLines.textlet chart: Chart = new Chart({ primaryXAxis: { stripLines: [{ text: 'stripline' }]}));chart.appendTo('#chart');</code>
Text alignment in stripline	<code>Property:stripLines.textAlignment\$("#chart").ejChart({ primaryXAxis: { stripLines: [{ textAlignment: 'Far; }]}));</code>	<code>Property:stripLines.horizontalAlignmentlet chart: Chart = new Chart({ primaryXAxis: { stripLines: [{ horizontalAlignment: 'Far' }]}));chart.appendTo('#chart');</code>
Vertical Text alignment in stripline	Not Applicable	<code>Property:stripLines.verticalAlignmentlet chart: Chart = new Chart({ primaryXAxis: { stripLines: [{ verticalAlignment: 'Far' }]}));chart.appendTo('#chart');</code>
Size of stripline	<code>Property:stripLines.width\$("#chart").ejChart({ primaryXAxis: { stripLines: [{ width: 10; }]}));</code>	<code>Property:stripLines.sizelet chart: Chart = new Chart({ primaryXAxis: { stripLines: [{ size: 10 }]}));chart.appendTo('#chart');</code>
ZIndex of stripline	<code>Property:stripLines.zIndex\$("#chart").ejChart({ primaryXAxis: { stripLines: [{ zIndex: 'Behind' }]}));</code>	<code>Property:stripLines.sizelet chart: Chart = new Chart({ primaryXAxis: { stripLines: [{ zIndex: 'Behind' }]}));chart.appendTo('#chart');</code>
Font style of stripline	<code>Property:stripLines.fontStyle\$("#chart").ejChart({ primaryXAxis: { stripLines: [{ fontStyle: {} }]}));</code>	<code>Property:stripLines.textStylelet chart: Chart = new Chart({ primaryXAxis: { stripLines: [{ textStyle: {} }]}));chart.appendTo('#chart');</code>

Multilevel Labels

<!-- markdownlint-disable MD033 -->

Behaviour	API in Essential JS 1	API in Essential JS 2
Default behaviour for multilevel Labels	<code>Property:primaryXAxis.multilevelLabels\$("#chart").ejChart({ primaryXAxis: { multilevelLabels: [{ visible: true }]}));</code>	<code>Property:primaryXAxis.multilevelLabelslet chart: Chart = new Chart({ primaryXAxis: { multilevelLabels: [{ }]}));chart.appendTo('#chart');</code>

Default behaviour for multilevel Labels	Property:primaryXAxis.multilevelLabels\$("#chart").ejChart({ primaryXAxis: { multilevelLabels: [{ visible: true }] } });	Property:primaryXAxis.multilevelLabels1 let chart: Chart = new Chart({ primaryXAxis: { multilevelLabels: [{ }] } });chart.appendTo('#chart');
Text alignment for multilevel Labels	Property:multiLevelLabels.textAlignment\$("#chart").ejChart({ primaryXAxis: { multilevelLabels: [{ textAlignment: 'Near' }] } });	Property:multilevelLabels.alignment let chart: Chart = new Chart({ primaryXAxis: { multilevelLabels: [{ alignment: 'Near' }] } });chart.appendTo('#chart');
Text overflow for multilevel Labels	Property:multiLevelLabels.textOverFlow\$("#chart").ejChart({ primaryXAxis: { multilevelLabels: [{ textOverFlow: 'Trim' }] } });	Property:multiLevelLabels.overFlow let chart: Chart = new Chart({ primaryXAxis: { multilevelLabels: [{ overFlow: 'Trim' }] } });chart.appendTo('#chart');
Border for multilevel Labels	Property:multiLevelLabels.border\$("#chart").ejChart({ primaryXAxis: { multilevelLabels: [{ border: { width: 2, color: 'red', type: 'brace' } }] } });	Property:multiLevelLabels.border let chart: Chart = new Chart({ primaryXAxis: { multilevelLabels: [{ border: { width: 2, color: 'red', type: 'brace' } }] } });chart.appendTo('#chart');
Start value for label	Property:multiLevelLabels.start\$("#chart").ejChart({ primaryXAxis: { multilevelLabels: [{ start: 45 }] } });	Property:multiLevelLabels.categories.start let chart: Chart = new Chart({ primaryXAxis: { multilevelLabels: [{ categories: [{ start: 45 }] }] } });chart.appendTo('#chart');
End value for label	Property:multiLevelLabels.start\$("#chart").ejChart({ primaryXAxis: { multilevelLabels: [{ end: 45 }] } });	Property:multiLevelLabels.categories.end let chart: Chart = new Chart({ primaryXAxis: { multilevelLabels: [{ categories: [{ end: 45 }] }] } });chart.appendTo('#chart');
Text for label	Property:multiLevelLabels.text\$("#chart").ejChart({ primaryXAxis: { multilevelLabels: [{ text: 'Start' }] } });	Property:multiLevelLabels.categories.text let chart: Chart = new Chart({ primaryXAxis: { multilevelLabels: [{ categories: [{ text: 'text' }] }] } });chart.appendTo('#chart');
maximum text width for label	Property:multiLevelLabels.maximumTextWidth\$("#chart").ejChart({ primaryXAxis: { multilevelLabels: [{ maximumTextWidth: 10 }] } });	Property:multiLevelLabels.categories.maximumTextWidth let chart: Chart = new Chart({ primaryXAxis: { multilevelLabels: [{ categories: [{

		<code>maximumTextWidth: 20 }} } }}}});chart.appendTo('#chart');</code>
level of labels	<code>Property:multiLevelLabels.level\$("#chart").ejChart({ primaryXAxis: { multilevelLabels: [{ level: 2 } }}}});</code>	Not applicable. Categories are used

Methods

<!-- markdownlint-disable MD033 -->

Behaviour	API in Essential JS 1	API in Essential JS 2
animation for series	<code>Property:chart.animate\$("#chart").ejChart({ animate: () { }});</code>	Not applicable
Redraw for chart	<code>Property:chart.redraw\$("#chart").ejChart({ redraw: () { }});</code>	<code>Property:chart.refresh()let chart: Chart = new Chart({});chart.appendTo('#chart');chart. width = '400';chart.refresh();</code>
Export	<code>Property:chart.export\$("#chart").ejChart({ export: () { }});</code>	<code>Property:chart.export()let chart: Chart = new Chart({});chart.export('JPEG', 'chart');chart.appendTo('#chart');</code>
Print	<code>Property:chart.print\$("#chart").ejChart({ print: () { }});</code>	<code>Property:chart.print()let chart: Chart = new Chart({});chart.print('chart');chart.appe ndTo('#chart');</code>
AddSeries	Not Applicable	<code>Property:chart.addSeries()let chart: Chart = new Chart({});chart.appendTo('#chart');chart. addSeries();</code>
RemoveSeries	Not Applicable	<code>Property:chart.removeSeries()let chart: Chart = new Chart({});chart.appendTo('#chart');chart. removeSeries();</code>

Events

<!-- markdownlint-disable MD033 -->

Behaviour	API in Essential JS 1	API in Essential JS 2
Fires on annotation click	<code>Property:annotationClick\$("#chart").ejChart({ annotationClick: () { }});</code>	Not applicable
Fires after animation	<code>Property:animationComplete\$("#chart").ejChart({ animationComplete: () { }});</code>	<code>Property:animationComplete()let chart: Chart = new Chart({ animationComplete: () => {</code>

		<code>}});chart.appendTo('#chart');</code>
Fires on axis label click	Property:axisLabelClick\$("#chart").ejChart({axisLabelClick: () { }});	Not applicable
Fires before axis label render	Property:axisLabelRendering\$("#chart").ejChart({axisLabelRendering: () { }});	Property:axisLabelRender() <code>let chart: Chart = new Chart({ axisLabelRender: () => { }});chart.appendTo('#chart');</code>
Fires on axis label mouseMove	Property:axisLabelMouseMove\$("#chart").ejChart({axisLabelMouseMove: () { }});	Not applicable
Fires on axis label initialize	Property:axisLabelInitialize\$("#chart").ejChart({axisLabelInitialize: () { }});	Not applicable
Fires before axis range calculation	Property:axesRangeCalculate\$("#chart").ejChart({axesRangeCalculate: () { }});	Property:axisRangeCalculated() <code>let chart: Chart = new Chart({ axisRangeCalculated: () => { }});chart.appendTo('#chart');</code>
Fires on axis title rendering	Property:axisTitleRendering\$("#chart").ejChart({axisTitleRendering: () { }});	Not applicable
Fires on after chart resize	Property:afterResize\$("#chart").ejChart({afterResize: () { }});	Not applicable
Fires on before chart resize	Property:beforeResize\$("#chart").ejChart({beforeResize: () { }});	Property:resized <code>let chart: Chart = new Chart({ resized: () => { }});chart.appendTo('#chart');</code>
Fires on chart click	Property:chartClick\$("#chart").ejChart({chartClick: () { }});	Property:chartMouseClicked <code>let chart: Chart = new Chart({ chartMouseClicked: () => { }});chart.appendTo('#chart');</code>
Fires on chart	Property:chartMouseMove\$("#chart").ejChart({chartMouseMove: () { }});	Property:chartMouseMove <code>let chart: Chart = new Chart({ chartMouseMove:</code>

mouse move		<pre>() => { }});chart.appendTo('#chart');</pre>
Fires on chart mouse leave	Property:chartMouseLeave <pre>\$("#chart").ejChart({ chartMouseLeave: () { }});</pre>	Property:chartMouseLeave <pre>let chart: Chart = new Chart({ chartMouseLeave: () => { }});chart.appendTo('#chart');</pre>
Fires on before chart double click	Property:chartDoubleClick <pre>\$("#chart").ejChart({ chartDoubleClick: () { }});</pre>	Not applicable
Fires on chart mouse up	Not Applicable	Property:chartmouseUp <pre>let chart: Chart = new Chart({ chartmouseUp: () => { }});chart.appendTo('#chart');</pre>
Fires on chart mouse down	Not Applicable	Property:chartmouseDown <pre>let chart: Chart = new Chart({ chartmouseDown: () => { }});chart.appendTo('#chart');</pre>
Fires during the calculation of chart area bounds. You can use this event to customize the bounds of chart area	Property:chartAreaBoundsCalculate <pre>\$("#chart").ejChart({ chartAreaBoundsCalculate: () { }});</pre>	Not applicable
Fires when the dragging is started	Property:dragStart <pre>\$("#chart").ejChart({ dragStart: () { }});</pre>	Not applicable
Fires while dragging	Property:dragging <pre>\$("#chart").ejChart({ dragging: () { }});</pre>	Not applicable

Fires when the dragging is completed	Property:dragEnd\$("#chart").ejChart({ dragEnd: () { } });	Property:dragCompletelet chart: Chart = new Chart({ dragComplete: () => { } });chart.appendTo('#chart');
Fires when chart is destroyed completely.	Property:destroy\$("#chart").ejChart({ destroy: () { } });	Not applicable
Fires after chart is created.	Property:create\$("#chart").ejChart({ create: () { } });	Property:loadedlet chart: Chart = new Chart({ loaded: () => { } });chart.appendTo('#chart');
Fires before rendering the data labels.	Property:displayTextRendering\$("#chart").ejChart({ displayTextRendering: () { } });	Property:textRenderlet chart: Chart = new Chart({ textRender: () => { } });chart.appendTo('#chart');
Fires, when error bar is rendering.	Property:errorBarRendering\$("#chart").ejChart({ errorBarRendering: () { } });	Not applicable
Fires during the calculation of legend bounds.	Property:legendBoundsCalculate\$("#chart").ejChart({ legendBoundsCalculate: () { } });	Not applicable
Fires on clicking the legend item.	Property:legendItemClick\$("#chart").ejChart({ legendItemClick: () { } });	Not applicable
Fires when moving mouse over legend item	Property:legendItemMouseMove\$("#chart").ejChart({ legendItemMouseMove: () { } });	Not applicable
Fires before rendering	Property:legendItemRendering\$("#chart").ejChart({ legendItemRendering: () { } });	Property:legendRenderlet chart: Chart = new Chart({ legendRender: () => {

the legend item.		}});chart.appendTo('#chart');
Fires before loading the chart.	Property:load\$("#chart").ejChart({ load: () { }});	Property:loadlet chart: Chart = new Chart({ load: () => { }});chart.appendTo('#chart');
Fires, when multi level labels are rendering.	Property:multiLevelLabelRendering\$("#chart").ejChart({ multiLevelLabelRendering: () { }});	Property:axisMultiLabelRenderlet chart: Chart = new Chart({ axisMultiLabelRender : () => { }});chart.appendTo('#chart');
Fires on clicking a point in chart.	Property:pointRegionClick\$("#chart").ejChart({ pointRegionClick: () { }});	Property:pointClicklet chart: Chart = new Chart({ pointClick : () => { }});chart.appendTo('#chart');
Fires when mouse is moved over a point.	Property:pointRegionMouseMove\$("#chart").ejChart({ pointRegionMouseMove: () { }});	Property:pointMovelet chart: Chart = new Chart({ pointMove : () => { }});chart.appendTo('#chart');
Fires before rendering chart.	Property:preRender\$("#chart").ejChart({ preRender: () { }});	Not applicable
Fires when point render.	Not Applicable	Property:pointRenderlet chart: Chart = new Chart({ pointRender : () => { }});chart.appendTo('#chart');
Fires after selected the data in chart.	Property:rangeSelected\$("#chart").ejChart({ rangeSelected: () { }});	Not applicable
Fires after selecting a series.	Property:seriesRegionClick\$("#chart").ejChart({ seriesRegionClick: () { }});	Not applicable
Fires before rendering a series.	Property:seriesRendering\$("#chart").ejChart({ seriesRendering: () { }});	Property:seriesRenderlet chart: Chart = new Chart({ seriesRender : () => {

		<code>}});chart.appendTo('#chart');</code>
Fires before rendering the marker symbols.	<code>Property:symbolRendering\$("#chart").ejChart({symbolRendering: () { }});</code>	Not applicable
Fires before rendering the trendline	<code>Property:trendlineRendering\$("#chart").ejChart({trendlineRendering: () { }});</code>	Not applicable
Fires before rendering the Chart title.	<code>Property:titleRendering\$("#chart").ejChart({titleRendering: () { }});</code>	Not applicable
Fires before rendering the Chart sub title.	<code>Property:subTitleRendering\$("#chart").ejChart({subTitleRendering: () { }});</code>	Not applicable
Fires before rendering the tooltip.	<code>Property:toolTipInitialize\$("#chart").ejChart({toolTipInitialize: () { }});</code>	<code>Property:tooltipRender let chart: Chart = new Chart({ tooltipRender : () => { }});chart.appendTo('#chart');</code>
Fires before rendering crosshair tooltip in axis	<code>Property:trackAxisToolTip\$("#chart").ejChart({trackAxisToolTip: () { }});</code>	Not applicable
Fires before rendering trackball tooltip.	<code>Property:trackToolTip\$("#chart").ejChart({trackToolTip: () { }});</code>	Not applicable
Event triggered when	<code>Property:scrollStart\$("#chart").ejChart({scrollStart: () { }});</code>	<code>Property:scrollStart let chart: Chart = new Chart({ scrollStart : () => { }});chart.appendTo('#chart');</code>

scroll starts.		
Event triggered when scroll ends.	Property:scrollEnd\$("#chart").ejChart({ scrollEnd: () { }});	Property:scrollEndlet chart: Chart = new Chart({ scrollEnd: () => { }});chart.appendTo('#chart');
Event triggered when scroll changes.	Property:scrollChange\$("#chart").ejChart({ scrollChange: () { }});	Property:scrollChangelet chart: Chart = new Chart({ scrollChange: () => { }});chart.appendTo('#chart');
Fires while performing rectangle zooming in chart.	Property:zoomComplete\$("#chart").ejChart({ zoomComplete: () { }});	Property:zoomCompletest chart: Chart = new Chart({ zoomComplete: () => { }});chart.appendTo('#chart'); ## Chart properties

<!-- markdownlint-disable MD033 -->

Behaviour	API in Essential JS 1	API in Essential JS 2
selected data index	Property:selectedDataPointIndexes\$("#chart").ejChart({ selectedDataPointIndexes: [{ seriesIndex: 0, pointIndex: 1}]});	Property:selectedDataIndexeslet chart: Chart = new Chart({selectedDataIndexes: [{ series: 0, point: 1}]});chart.appendTo('#chart');
sideBySideSeries Placement for column based series	Property:sideBySideSeriesPlacement\$("#chart").ejChart({ sideBySideSeriesPlacement});	Property:sideBySidePlacementlet chart: Chart = new Chart({ sideBySidePlacement: true});chart.appendTo('#chart');
ZoomSettings	Property:zooming\$("#chart").ejChart({ zooming: { enable: true, enableDeferredZoom: true, enablePinch: true, enableMouseWheel: true, enableScrollBar: true, toolbarItems: [], type: 'X' }});	Property:zoomSettingslet chart: Chart = new Chart({ zoomSettings: { enable: true, enablePinchZooming: true, enableDeferredZooming: true, enableMouseWheelZooming: true, enableSelectionZooming: true, enableScrollBar: true }});chart.appendTo('#chart');
Background color of the chart	Property:background\$("#container").ejChart({ background: 'transparent'});	Property:backgroundlet chart: Chart = new Chart({ background: '#EEFFCC'});chart.appendTo('#chart');

URL of the image to be used as chart background.	Property:backGroundImageUrl \$("#container").ejChart({ backGroundImageUrl : '../images/chart/wheat.png'}));	Not Applicable
Customizing border of the chart	Property:border \$("#container").ejChart({ border: { width: 2, color: '#CCEEFF', opacity: 0.5}});	Property:borderlet chart: Chart = new Chart({ border: { width: 2, color: '#CCEEFF'}});chart.appendTo('#char t');
This provides options for customizing export settings	Property:exportSettings\$("#containe r").ejChart({ exportSettings: { filename : "chart", angle: '45' }});	Property:export()let chart: Chart = new Chart({ border: { width: 2, color: '#CCEEFF'}});chart.appendTo('#char t');chart.export(type, fileName);
ChartArea customization	Property:chartArea\$("#container") .ejChart({ chartArea: { background: 'transparent', border: { opacity: 0.3, color: 'red', width: 2}}});	Property:chartArealet chart: Chart = new Chart({ chartArea: { background: 'transparent', border: { width: 2, color: '#CCEEFF' }});chart.appendTo('#chart');chart .export(type, fileName);

How To

Live chart in ##Platform_Name## Chart control

You can update a chart with live data by using the set interval.

To update live data in a chart, follow the given steps:

Step 1:

Initialize the chart with series.

```
import { Chart } from '@syncfusion/ej2-charts';
```

```
// initialize Chart component
```

```
let chart: Chart = new Chart(
```

```
//Initializing Chart Series
```

```
series:[
```

```
type: 'Line',
```

```
]
```

```
);
```

```
// render initialized Chart
```

```
chart.appendTo('#container');
```

```
`
```

Step 2:

Update the data to series, and refresh the chart at specified interval by using the set interval.

To refresh the chart, invoke the `refresh` method.

INDEX.JS

```
var series1 = [];
var value = 10;
var i;
var intervalId;
for (i = 0; i < 50; i++) { // Data update
    if (Math.random() > .5) {
        value += Math.random() * 2.0;
    }
    series1[i] = { x: i, y: value };
}
var chart = new ej.charts.Chart({
    series: [
        {
            type: 'Line',
            dataSource: series1,
            xName: 'x',
            yName: 'y', animation: { enable: false }
        },
    ],
    width: '650px',
    height: '350px'
}, '#element');
var setTimeoutValue = 100;
intervalId = setInterval(
    function() {
        if (document.getElementById('container') === null) {
            clearInterval(intervalId);
        } else {
            if (Math.random() > .5) {
                value += Math.random() * 2.0;
            }
            series1.push({ x: i, y: value });
            i++;
            series1.shift(); // Used to remove the first element
            chart.series[0].dataSource = series1;
            chart.refresh();
        }
    }, setTimeoutValue);
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div class="col-sm-8">
        <div class="row">
            <div class="col-sm-4">
                <div id="container">
                    <div id="element" style="width:350px;
height:350px;float:left">
                </div>
                <label id="lbl"></label>
            </div>
            <div class="col-sm-4" style="width:200px;
height:350px;float:right">
                <div id="Grid">
                </div>
            </div>
        </div>
    </div>

<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Prevent data label in ##Platform_Name## Chart control

To prevent the chart data label when the data value is 0, follow the given steps:

Step 1:

Get the point value and check whether the `args.point.y` value is zero or not by using the [textRender](#) event. If the value is zero, then set the `args.cancel` to true.

The output will appear as follows,

INDEX.JS

```

var chart = new ej.charts.Chart({
    //Initializing Primary X Axis
    primaryXAxis: {
        valueType: 'DateTime',
    },
    //Initializing Chart Series
    series: [
        {
            type: 'Line',
            dataSource: [

```

```

        { x: new Date(2005, 0, 1), y: 21 }, { x: new Date(2006,
0, 1), y: 24 },
        { x: new Date(2007, 0, 1), y: 0 }, { x: new Date(2008,
0, 1), y: 38 },
        { x: new Date(2009, 0, 1), y: 54 }, { x: new Date(2010,
0, 1), y: 57 },
    ],
    xName: 'x', width: 2, marker: {
        dataLabel : { visible: true },
        visible: true,
        width: 10,
        height: 10
    },
    yName: 'y', name: 'Germany',
}
},
//Initializing Chart title
title: 'Inflation - Consumer Price',
textRender: function(args) {
    args.cancel = args.point.y === 0;
},
width: '650px',
height: '350px'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div class="col-sm-8">
        <div class="row">
            <div class="col-sm-4">
                <div id="container">
                    <div id="element" style="width:350px;
height:350px;float:left">
                </div>
                <label id="lbl"></label>
            </div>

```

```

        </div>
        <div class="col-sm-4" style="width:200px;
height:350px;float: right">
            <div id="Grid">
                </div>
            </div>
        </div>
    </div>

<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Tool tip format in ##Platform_Name## Chart control

Using [tooltipRender](#) event, you can able to format the datetime value instead of rendered value.

To format the datetime value, please follow the steps below

Step 1:

By using [tooltipRender](#) event we can able to get the current point x value. Using this value to format the tooltip by using `formatDate` method.

The output will appear as follows,

INDEX.JS

```

var chart = new ej.charts.Chart({
    //Initializing Primary X Axis
    primaryXAxis: {
        valueType: 'DateTime',
    },
    //Initializing Chart Series
    series: [
        {
            type: 'Line',
            dataSource: [
                { x: new Date(2005, 0, 1), y: 21 }, { x: new Date(2006,
0, 1), y: 24 },
                { x: new Date(2007, 0, 1), y: 30 }, { x: new Date(2008,
0, 1), y: 38 },
                { x: new Date(2009, 0, 1), y: 54 }, { x: new Date(2010,
0, 1), y: 57 },
            ],
            xName: 'x', width: 2, marker: {
                dataLabel : { visible: true },
                visible: true,
                width: 10,
                height: 10
            },
            yName: 'y', name: 'Germany',
        }
    ]
});

```



```

    ],
    //Initializing Chart title
    title: 'Inflation - Consumer Price',
    tooltip: {enable: true},
    tooltipRender: function(args) { // To format the current point x
value
        var intl = new ej.base.Internationalization();
        var formattedString = intl.formatDate(new Date(args.point.x), {
skeleton: 'yMd' });
        args.text = formattedString;
    },
    width: '650px',
    height: '350px'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div class="col-sm-8">
        <div class="row">
            <div class="col-sm-4">
                <div id="container">
                    <div id="element" style="width:350px;
height:350px;float:left">
                </div>
                <label id="lbl"></label>
            </div>
            <div class="col-sm-4" style="width:200px;
height:350px;float: right">
                <div id="Grid">
                </div>
            </div>
        </div>
    </div>

</script>

```

```

var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Add series in ##Platform_Name## Chart control

You can add or remove the chart series dynamically by using the `addSeries` or `removeSeries` method.

To add or remove the series dynamically, follow the given steps:

Step 1:

To add a new series to chart dynamically, pass the series value to the `addSeries` method.

To remove the new series from chart dynamically, pass the series index to the `removeSeries` method.

INDEX.JS

```

var chart = new ej.charts.Chart({
    primaryXAxis: {
        title: 'Manager',
        valueType: 'Category',
    },
    //Initializing Primary Y Axis
    primaryYAxis: {
        title: 'Sales',
        minimum: 0,
        maximum: 20000,
    },
    //Initializing Chart Series
    series: [
        {
            type: 'Column',
            dataSource: [{ x: 'John', y: 10000 }, { x: 'Jake', y: 12000 }, { x: 'Peter', y: 18000 }, { x: 'James', y: 11000 }, { x: 'Mary', y: 9700 }],
            xName: 'x', width: 2,
            yName: 'y'
        }
    ],
    //Initializing Chart title
    title: 'Sales Comparision',
}, '#element');
document.getElementById('add').onclick = () => {
    chart.addSeries([
        {
            type: 'Column',
            dataSource: [{ x: 'John', y: 11000 }, { x: 'Jake', y: 16000 }, { x: 'Peter', y: 19000 }, { x: 'James', y: 12000 }, { x: 'Mary', y: 10700 }],
            xName: 'x', width: 2,
            yName: 'y'
        }
    ]);
};

```

```
document.getElementById('remove').onclick = () => {
    chart.removeSeries(1);
};
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element" style="height: 300px"></div>
    <button id="add" type="button" width="15%" style="float:
left">Add</button>
    <button id="remove" type="button" width="15%" style="float:
right">Remove</button>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Points customization in ##Platform_Name## Chart control

You can customize the series points with patterns by using the `pointColorMapping` property.

To customize the series point colors, follow the given steps:

Step 1:

Define the patterns and map the pattern URL to the series point using `pointColorMapping` property in series.

INDEX.TS

```
import { Chart, ColumnSeries, Category, DataLabel, Tooltip } from
'@syncfusion/ej2-charts';
Chart.Inject(ColumnSeries, DataLabel, Category, Tooltip);
import { EmitType } from 'syncfusion/ej2-base';
let chart: Chart = new Chart({
  //Initializing Primary X Axis
```

```

    primaryXAxis: {
        valueType: 'Category',
    },
    //Initializing Primary Y Axis
    primaryYAxis:
    {
        minimum: 0, maximum: 250, interval: 50
    },
    //Initializing Chart Series
    series: [
        {
            type: 'Column', xName: 'x', width: 2, yName: 'y',
pointColorMapping: 'color',
            dataSource: [
                { x: 'BGD', y: 106, text: 'Bangaladesh', color:
'url(#chess)' },
                { x: 'BTN', y: 103, text: 'Bhutn', color:
'url(#cross)' },
                { x: 'NPL', y: 198, text: 'Nepal', color:
'url(#circle)' },
                { x: 'THA', y: 189, text: 'Thiland', color:
'url(#rectangle)' },
                { x: 'MYS', y: 230, text: 'Malaysia', color:
'url(#line)' }
            ], name: 'Tiger',
            cornerRadius: {
                bottomLeft: 10, bottomRight: 10, topLeft: 10, topRight:
10
            }
        }
    ],
    legendSettings: { visible: false },
    //Initializing Chart title
    title: 'Tiger Population - 2016',
    width: '650px',
    height: '300px'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```

```

<svg width="20" height="20" xmlns="http://www.w3.org/2000/svg"
version="1.1">
  <defs>
    <pattern id="chess" x="0" y="0"
patternUnits="userSpaceOnUse" width="20" height="20">
      <rect width="20" height="20"
fill="#fff"></rect>
      <rect width="10" height="10"
fill="#ed7d31"></rect>
      <rect x="10" y="10" width="10" height="10"
fill="#ed7d31"></rect>
    </pattern>
    <pattern id="cross" x="0" y="0"
patternUnits="userSpaceOnUse" width="8" height="8">
      <rect width="8" height="8"
fill="#4472c4"></rect>
      <path d="M0 0L8 8Z" stroke-width="1"
stroke="white"></path>
    </pattern>
    <pattern id="circle" x="0" y="0"
patternUnits="userSpaceOnUse" width="9" height="9">
      <rect fill="#FFFFFF" width="9"
height="9"></rect>
      <circle fill="#f7ce69" cx="5.125" cy="3.875"
r="3.625"></circle>
    </pattern>
    <pattern id="rectangle" x="0" y="0"
patternUnits="userSpaceOnUse" width="12" height="12">
      <rect fill="#FFFFFF" width="12"
height="11"></rect>
      <rect x="1" y="2" fill="#404041" width="4"
height="9"></rect>
      <rect x="7" y="2" fill="#404041" width="4"
height="9"></rect>
    </pattern>
    <pattern id="line" x="0" y="0"
patternUnits="userSpaceOnUse" width="12" height="12">
      <line fill="none" stroke="#7ddf1e" stroke-
miterlimit="10" x1="0" y1="1.5" x2="10" y2="1.5"></line>
      <line fill="none" stroke="#7ddf1e" stroke-
miterlimit="10" x1="0" y1="5.5" x2="10" y2="5.5"></line>
      <line fill="none" stroke="#7ddf1e" stroke-
miterlimit="10" x1="0" y1="9.5" x2="10" y2="9.5"></line>
    </pattern>
  </defs>
</svg>

<div id="container">
  <div id="element" style="height: 300px"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>

```

```
</body></html>
```

Stacking total in ##Platform_Name## Chart control

By using the [annotation](#), you can show any element in desired view.

To show the total value in data points, follow the given steps:

Step 1:

Define annotation for each x point in chart, now change the annotation value in chart by using the [annotationRender](#) event. In this event, assign the stacked value of the last series to the annotation to show the total value of the

stacking series.

INDEX.JS

```
var i = 0;
var chart = new ej.charts.Chart({
  primaryXAxis: {
    valueType: 'Category', interval: 1
  },
  // Initialize the chart series
  series: [
    {
      type: 'StackingColumn', xName: 'x', width: 2, yName: 'y',
      name: 'Apple', animation: {enable: false},
      dataSource: [{ x: 'Jamesh', y: 5 }, { x: 'Michael', y: 4 },
        { x: 'John', y: 5 }],
      marker: { dataLabel: { visible: true, position: 'Top', font:
        { fontWeight: '600', color: '#ffffff' } } }
    }, {
      type: 'StackingColumn', xName: 'x', width: 2, yName: 'y',
      name: 'Orange', animation: {enable: false},
      dataSource: [{ x: 'Jamesh', y: 4 }, { x: 'Michael', y: 3 },
        { x: 'John', y: 4 }],
      marker: { dataLabel: { visible: true, position: 'Top', font:
        { fontWeight: '600', color: '#ffffff' } } }
    }, {
      type: 'StackingColumn', xName: 'x', width: 2, yName: 'y',
      name: 'Grapes', animation: {enable: false},
      dataSource: [{ x: 'Jamesh', y: 1 }, { x: 'Michael', y: 2 },
        { x: 'John', y: 2 }],
      marker: { dataLabel: { visible: true, position: 'Top', font:
        { fontWeight: '600', color: '#ffffff' } } }
    }
  ],
  annotations:[
    {
      content: '<div id="point1" style="font-size:11px;font-weight:bold;color:gray;fill:gray;"><span>12</span></div>',
      x: 'Jamesh', y: '11', coordinateUnits: 'Point', region:
      'Series'
    },
    {

```

```

        content: '<div id="point1" style="font-size:11px;font-weight:bold;color:gray;fill:gray;"><span>12</span></div>',
        x: 'Michael', y: '10', coordinateUnits: 'Point', region:
'Series'
    },
    {
        content: '<div id="point1" style="font-size:11px;font-weight:bold;color:gray;fill:gray;"><span>12</span></div>',
        x: 'John', y: '12', coordinateUnits: 'Point', region:
'Series'
    }
],
// Initialize the chart title
title: 'Fruit Consumption', tooltip: { enable: true, shared: true },
annotationRender: function(args) {
    var length = args.chart.series.length - 1;
    var value = args.chart.series[length].stackedValues.endValues[i];
    i += (i == length) ? -length : 1;
    args.content.children[0].children[0].innerHTML = value;
},
width: '650px',
height: '350px'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet">
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div class="col-sm-8">
    <div class="row">
      <div class="col-sm-4">
        <div id="container">
          <div id="element" style="width:350px;
height:350px;float:left">
          </div>
          <label id="lbl"></label>
        </div>
      </div>
    </div>
  </div>

```

```

        <div class="col-sm-4" style="width:200px;
height:350px;float: right">
            <div id="Grid">
                </div>
            </div>
        </div>
    </div>

<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Selected data grid in ##Platform_Name## Chart control

By using the [dragComplete](#), you can get the selected data values for range selection.

To display the selected data value, follow the given steps:

Step 1:

Get the selected data point values and display the values through grid component by using the [dragComplete](#) event.

INDEX.JS

```

var chart = new ej.charts.Chart({
    primaryXAxis: {
        minimum: 1970,
        maximum: 2016
    },
    //Initializing Primary Y Axis
    primaryYAxis: {
        title: 'Sales',
        labelFormat: '{value}%',
        interval: 25,
        minimum: 0,
        maximum: 100,
    },
    //Initializing Chart Series
    series: [
        {
            type: 'Scatter',
            dataSource: [{ x: 1971, y: 50 }, { x: 1972, y: 20 }, { x: 1973,
y: 63 }, { x: 1974, y: 81 }, { x: 1975, y: 64 },
            { x: 1976, y: 36 }, { x: 1977, y: 22 }, { x: 1978, y: 78 }, { x:
1979, y: 60 }, { x: 1980, y: 41 },
            { x: 1981, y: 62 }, { x: 1982, y: 56 }, { x: 1983, y: 96 }, { x:
1984, y: 48 }, { x: 1985, y: 23 },
            { x: 1986, y: 54 }, { x: 1987, y: 73 }, { x: 1988, y: 56 }, { x:
1989, y: 67 }, { x: 1990, y: 79 },
            { x: 1991, y: 18 }, { x: 1992, y: 78 }, { x: 1993, y: 92 }, { x:
1994, y: 43 }, { x: 1995, y: 29 },

```



```

    { x: 1996, y: 14 }, { x: 1997, y: 85 }, { x: 1998, y: 24 }, { x:
1999, y: 61 }, { x: 2000, y: 80 },
    { x: 2001, y: 14 }, { x: 2002, y: 34 }, { x: 2003, y: 81 }, { x:
2004, y: 70 }, { x: 2005, y: 21 },
    { x: 2006, y: 70 }, { x: 2007, y: 32 }, { x: 2008, y: 43 }, { x:
2009, y: 21 }, { x: 2010, y: 63 },
    { x: 2011, y: 9 }, { x: 2012, y: 51 }, { x: 2013, y: 25 }, { x:
2014, y: 96 }, { x: 2015, y: 32 }
    ],
    xName: 'x',
    yName: 'y', name: 'Product A',
    marker: {
        shape: 'Triangle',
        width: 10, height: 10
    }
    },
    title: 'Profit Comparision of A and B', legendSettings: { visible: true,
toggleVisibility: false },
    selectionMode: 'DragXY',
    //Get selected range data values
    dragComplete: function (args) {
        grid.dataSource = args.selectedDataValues[0];
        grid.refresh();
    }
}, '#element');
var grid = new ej.grids.Grid({
    columns: [
        { field: 'x', headerText: 'x', type: 'string' },
        { field: 'y', headerText: 'y', type: 'number' }
    ],
}, '#Grid');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <div class="container">
        <div id="Grid"></div>
```

```

<div id="element"></div>
</div>

<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Marker customization in ##Platform_Name## Chart control

By using the [pointRender](#), you can customize the marker shape.

To Customize the marker shape, follow the given steps:

Step 1:

Customize the marker shape in each data point by using the [pointRender](#) event. Using this event, you can set the `shape` value to the argument.

INDEX.JS

```

var shapes = [
    'Diamond', 'Circle', 'Rectangle', 'Line', 'Triangle', 'Rectangle'
];
var shapeRender = function(args) {
    args.shape = shapes[args.point.index];
}
var chart = new ej.charts.Chart({
    //Initializing Primary X Axis
    primaryXAxis: {
        title: 'Countries', valueType: 'Category',
        interval: 1
    },
    //Initializing Primary Y Axis
    primaryYAxis: {
        title: 'Penetration', rangePadding: 'None',
        labelFormat: '{value}%', minimum: 0,
        maximum: 75, interval: 15
    },
    //Initializing Chart Series
    series: [
        {
            type: 'Line',
            dataSource: [{ x: 'WW', y: 12, text: 'World Wide' },
                { x: 'EU', y: 5, text: 'Europe' },
                { x: 'APAC', y: 15, text: 'Pacific' },
                { x: 'LATAM', y: 6.4, text: 'Latin' },
                { x: 'MEA', y: 30, text: 'Africa' },
                { x: 'NA', y: 25.3, text: 'America' }],
            name: 'December 2007',
            marker: {
                visible: true, width: 10, height: 10,
                shape: 'Diamond', dataLabel: { name: 'text' }
            }
        }
    ]
});

```

```

        },
        xName: 'x', width: 2,
        yName: 'y',
    },
    ],
    //Initializing Chart title
    title: 'FB Penetration of Internet Audience',
    legendSettings: { visible: false },
    pointRender: shapeRender,
    width: '650px',
    height: '350px'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet">
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div class="col-sm-8">
    <div class="row">
      <div class="col-sm-4">
        <div id="container">
          <div id="element" style="width:350px;
height:350px;float:left">
            </div>
          <label id="lbl"></label>
        </div>
      </div>
      <div class="col-sm-4" style="width:200px;
height:350px;float: right">
        <div id="Grid">
          </div>
        </div>
      </div>
    </div>
  </div>

  <script>
var ele = document.getElementById('container');
if(ele) {

```

```

    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Legend customization in ##Platform_Name## Chart control

By using the [legendRender](#), you can customize the legend shape.

To Customize the legend shape, follow the given steps:

Step 1:

Set the shape value for each legend using `args.shape` in [legendRender](#) event.

INDEX.JS

```

var chart = new ej.charts.Chart({
  primaryXAxis: {
    valueType: 'Double',
  },
  //Initializing Primary Y Axis
  primaryYAxis: {
    title: 'Production (Billion as kWh)',
    valueType: 'Double'
  },
  //Initializing Chart Series
  series: [
    {
      type: 'StepArea',
      dataSource: [{ x: 2000, y: 416 }, { x: 2001, y: 490 }, { x:
2002, y: 470 }, { x: 2003, y: 500 },
      { x: 2004, y: 449 }, { x: 2005, y: 470 }, { x: 2006, y: 437
}, { x: 2007, y: 458 }],
      name: 'Renewable',
      xName: 'x', width: 2,
      yName: 'y',
    },
    {
      type: 'StepArea',
      dataSource: [{ x: 2000, y: 180 }, { x: 2001, y: 240 }, { x:
2002, y: 370 }, { x: 2003, y: 200 },
      { x: 2004, y: 229 }, { x: 2005, y: 210 }, { x: 2006, y: 337
}, { x: 2007, y: 258 }],
      name: 'Non-Renewable',
      xName: 'x', width: 2,
      yName: 'y',
    },
  ],
  //Initializing Chart title
  title: 'Electricity- Production',
  legendRender: function(args) {
    if (args.text === 'Renewable') {
      args.shape = 'Circle';
    } else if (args.text === 'Non-Renewable') {
      args.shape = 'Triangle';
    }
  }
});

```

```

    },
    width: '650px',
    height: '350px'
  }, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet">
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div class="col-sm-8">
    <div class="row">
      <div class="col-sm-4">
        <div id="container">
          <div id="element" style="width:350px;
height:350px;float:left">
          </div>
          <label id="lbl"></label>
        </div>
      </div>
      <div class="col-sm-4" style="width:200px;
height:350px;float: right">
        <div id="Grid">
          </div>
        </div>
      </div>
    </div>

  </div>

  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Tool tip table in ##Platform_Name## Chart control

You can show the tooltip as table by using template property in tooltip.

Follow the given steps to show the table tooltip,

Step 1:

Initialize the tooltip template div as shown in the following html page,

,

```
<div id='templateWrap'>
```

Female	
#{x}:	#{y}

```
</div>
```

,

Step 2:

To show that tooltip template, set the element id to the `template` property in tooltip.

INDEX.JS

```
var chart = new ej.charts.Chart({
  title: 'Population of India ( 2010 - 2016 )',
  // Initialize the chart axes
  primaryXAxis: {
    minimum: 2010, maximum: 2016,
    edgeLabelPlacement: 'Shift',
  },
  primaryYAxis: {
    minimum: 900, maximum: 1300,
    labelFormat: '{value}M',
  },
  // Initialize the chart series
  series: [
    {
      name: 'Female',
      dataSource: [
        { x: 2010, y: 990 }, { x: 2011, y: 1010 },
        { x: 2012, y: 1030 }, { x: 2013, y: 1070 },
        { x: 2014, y: 1105 }, { x: 2015, y: 1138 },
        { x: 2016, y: 1155 }
      ], xName: 'x', yName: 'y',
      marker: {
        visible: true,
        shape: 'Rectangle',
        width: 2
      }
    }
  ],
  tooltip: {
    enable: true,
    template: '#Female-Material'
```

```

    },
    width: '650px',
    height: '350px'
  }, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet">
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script id="Female-Material" type="text/x-template">
    <div id='templateWrap'>
      <table style="width:100%; border: 1px solid black;">
        <tr><th colspan="2" bgcolor="#00FFFF">Female</th></tr>
        <tr><td bgcolor="#00FFFF">${x}:</td><td
bgcolor="#00FFFF">${y}</td></tr>
      </table>
    </div>
  </script>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Footer in ##Platform_Name## Chart control

By using **annotation**, you can place any html elements to chart in a desired view.

To create footer and watermark for chart, follow the given steps:

Step 1:

Initialize the custom elements by using the **annotation** property.

By using the `content` option of the annotation object, you can specify the id of the element that needs to be displayed in the chart area as follow,

INDEX.JS

```
var chart = new ej.charts.Chart({
    //Initializing Primary X Axis
    primaryXAxis: {
        valueType: 'Category',
        interval: 1, majorGridLines: { width: 0 },
        labelIntersectAction: 'Rotate90'
    },
    //Initializing Annotations
    annotations: [{ // watermark for chart
        content: '<div id="chart_cloud" style="font-size:450%; opacity:
0.3;" >syncfusion</div>',
        x: 'Wed', y: 20, coordinateUnits: 'Point', horizontalAlignment:
'Center'
    }, { //footer for chart
        content: '<div id="chart" > <a href="https://www.syncfusion.com"
target="_blank">www.syncfusion.com</a></div>',
        x: 400, y: 340, coordinateUnits: 'Pixel', horizontalAlignment:
'Center'
    }
    ],
    //Initializing Primary Y Axis
    primaryYAxis:
    {
        minimum: 0,
        maximum: 40,
        interval: 10,
    },
    //Initializing Chart Series
    series: [
        {
            type: 'Spline',
            dataSource: [
                { x: 'Sun', y: 15 }, { x: 'Mon', y: 5 }, { x: 'Tue', y:
32 },
                { x: 'Wed', y: 15 }, { x: 'Thu', y: 29 }, { x: 'Fri', y:
24 },
                { x: 'Sat', y: 18 },
            ],
            xName: 'x', width: 2, marker: {
                visible: true
            },
            yName: 'y', name: 'Max Temp',
        }
    ],
    //Initializing Chart title
    title: 'NC Weather Report - 2016',
    width: '650px',
    height: '350px'
}, '#element');
```

INDEX.HTML


```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet">
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script id="Female-Material" type="text/x-template">
    <div id='templateWrap'>
      <table style="width:100%; border: 1px solid black;">
        <tr><th colspan="2" bgcolor="#00FFFF">Female</th></tr>
        <tr><td bgcolor="#00FFFF">${x}</td><td
bgcolor="#00FFFF">${y}</td></tr>
      </table>
    </div>
  </script>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Threshold in ##Platform_Name## Chart control

You can mark a threshold in chart by using the **stripline**.

To mark a threshold in chart, follow the given steps:

Step 1:

By using the start and end properties of **striplines** object in vertical axis, you can mark the threshold for y values of the series.

INDEX.JS

```

var chart = new ej.charts.Chart({
  //Initializing Primary X Axis
  primaryXAxis: {
    valueType: 'Category',

```

```

    },
    //Initializing Primary Y Axis
    primaryYAxis:
    {
        minimum: 10, maximum: 40, interval: 5,
        lineStyle: { color: '#808080' }, labelFormat: '{value} °C',
        rangePadding: 'None',
        //Initializing Striplines
        stripLines: [
            {
                start: 30, end: 30.1, color: '#ff512f', visible: true,
                textStyle: { size: '18px', color: '#ffffff', fontWeight:
'600' },
            }
        ]
    },
    //Initializing Chart Series
    series: [
        {
            dataSource: [
                { x: 'Sun', y: 28 }, { x: 'Mon', y: 27 }, { x: 'Tue', y:
33 }, { x: 'Wed', y: 36 },
                { x: 'Thu', y: 28 }, { x: 'Fri', y: 30 }, { x: 'Sat', y:
31 }],
            xName: 'x', width: 2, yName: 'y', type: 'Line', name:
'Weather',
            marker: { visible: true, width: 10, height: 10, border: {
width: 2, color: '#ffffff' }, fill: '#666666' },
        },
        {
            legendSettings: { visible: false },
            //Initializing Chart Title
            title: 'Weather Report',
            width: '650px',
            height: '350px'
        },
    ],
    '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>

```

```

</head>
<body>

    <div class="col-sm-8">
        <div class="row">
            <div class="col-sm-4">
                <div id="container">
                    <div id="element" style="width:350px;
height:350px;float:left">
                </div>
                <label id="lbl"></label>
            </div>
            <div class="col-sm-4" style="width:200px;
height:350px;float: right">
                <div id="Grid">
                </div>
            </div>
        </div>
    </div>

    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Grid data chart in ##Platform_Name## Chart control

You can visualize the data that returned by grid in chart.

To visualize the data in chart, follow the given steps:

Step 1:

Initialize the grid with datasource.

Step 2:

By using the grid's `actionComplete` event and `getCurrentViewRecords` method, you can get the current page records.

By using the grid's `databound` event, you can update the current page records into the chart's datasource and visualize the grid data in chart.

INDEX.JS

```

var chart = new ej.charts.Chart();
var data = new ej.data.DataManager(orderData).executeLocal(new
ej.data.Query().take(100));
var grid = new ej.grids.Grid({
    dataSource: data,
    allowPaging: true,
    pageSettings: { pageSize: 10 },
    columns: [

```

```

        { field: 'OrderDate', headerText: 'Order Date', width: 130, format:
'yMd', textAlign: 'Right' },
        { field: 'Freight', width: 120, format: 'C2', textAlign: 'Right' }
    ],
    dataBound: function () {
        chart = new ej.charts.Chart({
            //Initializing Primary X Axis
            primaryXAxis: {
                valueType: 'DateTime',
                intervalType: 'Days'
            },
            series: [
                {
                    type: 'Line',
                    dataSource: grid.getCurrentViewRecords(),
                    xName: 'OrderDate', width: 2, marker: {
                        visible: true
                    },
                    yName: 'Freight', name: 'Germany'
                },
            ],
        }, '#Chart');
    },
    actionComplete: function (args) {
        if (args.requestType === 'paging') {
            chart.series[0].dataSource = grid.getCurrentViewRecords();
            chart.refresh();
        }
    }
}, '#Grid');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/material.css" rel="stylesheet">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <div class="container">
        <div id="Grid"></div>
        <div id="Chart"></div>
    </div>

```

```

<script>
    var ele = document.getElementById('container');
    if (ele) {
        ele.style.visibility = "visible";
    }
</script>
<script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Percentage tool tip in ##Platform_Name## Chart control

By using the [tooltipRender](#) event, you can show the percentage value for each point of pie series in tooltip.

To show the percentage value in pie tooltip, follow the given steps:

Step 1:

By using the [tooltipRender](#) event, you can get the `args.point.y` and `args.series.sumOfPoints` values. You can use these values to calculate the percentage value for each point of pie series. To display the percentage value in tooltip, use the `args.content` property.

INDEX.JS

```

var chart = new ej.charts.AccumulationChart({
    series: [
        {
            dataSource: [
                { 'x': 'Chrome', y: 37 }, { 'x': 'UC Browser', y: 17 },
                { 'x': 'iPhone', y: 19 }, { 'x': 'Others', y: 4, text:
'4%' }, { 'x': 'Opera', y: 11 }
            ],
            dataLabel: {
                visible: true
            },
            radius: '70%', xName: 'x',
            yName: 'y', startAngle: 0,
            endAngle: 360, innerRadius: '0%'
        }
    ],
    enableSmartLabels: true,
    legendSettings: {
        visible: false,
    },
    // Initialize the tooltip
    tooltip: { enable: true },
    title: 'Mobile Browser Statistics',
    tooltipRender: function(args) {
        var value = args.point.y / args.series.sumOfPoints * 100;
        args.text = args.point.x + ' ' + Math.ceil(value) + ' %';
    },
    width: '650px',
    height: '350px'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet">
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div class="col-sm-8">
    <div class="row">
      <div class="col-sm-4">
        <div id="container">
          <div id="element" style="width:350px;
height:350px;float:left">
            </div>
            <label id="lbl"></label>
          </div>
        </div>
        <div class="col-sm-4" style="width:200px;
height:350px;float: right">
          <div id="Grid">
            </div>
          </div>
        </div>
      </div>
    </div>

  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Data label template in ##Platform_Name## Chart control

You can bind text and interior information for a point from dataSource other than x and y value. To change color for the background in the datalabel template, you can use `${point.text}`.

To use point.text, you have to bind the property from dataSource to name in the datalabel options.

Follow the given steps to show the table tooltip,

Step 1:

Initialize the datalabel template div as shown in the following html page,

```
<script id="index" type="text/x-template">
<div id='templateWrap' style="background-color: ${point.text}; border-radius:
3px;"><span>${point.y}</span></div>
</script>
```

Step 2:

To show that datalabel template, set the element id to the `template` property in datalabel.

INDEX.JS

```
var datalabelData = [
  { x: 10, y: 7000, color: 'red' },
  { x: 20, y: 1000, color: 'yellow' },
  { x: 30, y: 12000, color: 'orange' },
  { x: 40, y: 14000, color: 'skyblue' },
  { x: 50, y: 11000, color: 'blue' },
  { x: 60, y: 5000, color: 'green' },
  { x: 70, y: 7300, color: 'pink' },
  { x: 80, y: 9000, color: 'white' },
  { x: 90, y: 12000, color: 'magenta' },
  { x: 100, y: 14000, color: 'purple' },
  { x: 110, y: 11000, color: 'teal' },
  { x: 120, y: 5000, color: 'gray' },
];
var Chart = new ej.charts.Chart({
  series:[{
    dataSource: datalabelData,
    xName: 'x', yName: 'y',
    type: 'Line',
    marker: { visible: true, dataLabel: { visible: true, name: 'color',
template: '#index'}}
  }],
  }, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet">
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
    <script id="index" type="text/x-template">
        <div id='templateWrap' style="background-color: ${point.text}; border-
radius: 3px;"> <span>${point.y}</span></div>
    </script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Hide tool tip in ##Platform_Name## Chart control

By using the [tooltipRender](#) event, you can cancel the tooltip for unselected series in the chart.

To hide the tooltip value in unselected series, follow the given steps:

Step 1:

By using the [tooltipRender](#) event, you can get the series elements in the arguments. By using this argument we can compare whether seriesElementclasslist is deselected container or not. If it is true then we cancel the tooltip by setting the value for `args.cancel` as true.

INDEX.JS

```

var chart = new ej.charts.Chart({
    //Initializing Primary X Axis
    primaryXAxis: {
        valueType: 'DateTime',
    },
    //Initializing Chart Series
    series: [
        {
            type: 'Line',
            dataSource: [
                { x: new Date(2005, 0, 1), y: 21 }, { x: new Date(2006,
0, 1), y: 24 },
                { x: new Date(2007, 0, 1), y: 36 }, { x: new Date(2008,
0, 1), y: 38 },
                { x: new Date(2009, 0, 1), y: 54 }, { x: new Date(2010,
0, 1), y: 57 },
            ],
            xName: 'x', width: 2, marker: {
                dataLabel : { visible: true },
            }
        }
    ]
});

```



```

        visible: true,
        width: 10,
        height: 10
    },
    yName: 'y', name: 'Germany',
},
{
    type: 'Line',
    dataSource: [
        { x: new Date(2005, 0, 1), y: 28 }, { x: new Date(2006,
0, 1), y: 44 },
        { x: new Date(2007, 0, 1), y: 48 }, { x: new Date(2008,
0, 1), y: 50 },
        { x: new Date(2009, 0, 1), y: 66 }, { x: new Date(2010,
0, 1), y: 78 }
    ],
    xName: 'x', width: 2, marker: {
        dataLabel : { visible: true },
        visible: true,
        width: 10,
        height: 10
    },
    yName: 'y', name: 'India',
}
],
//Initializing Chart title
title: 'Inflation - Consumer Price',
selectionMode: 'Series',
tooltip: { enable: true },
tooltipRender: function(args) {
    var series = args.series;
    if (series.seriesElement.classList[0] === 'element_ej2_deselected') {
        args.cancel = true;
    }
},
width: '650px',
height: '350px'
}, '#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div class="col-sm-8">
        <div class="row">
            <div class="col-sm-4">
                <div id="container">
                    <div id="element" style="width:350px;
height:350px;float:left">
                </div>
                <label id="lbl"></label>
            </div>
            <div class="col-sm-4" style="width:200px;
height:350px;float: right">
                <div id="Grid">
                </div>
            </div>
        </div>
    </div>

<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Dotted line in ##Platform_Name## Chart control

By using **annotation**, you can add dotted lines in the chart.

To add dotted lines in the chart, follow the given steps:

Step 1:

Initialize the custom elements by using the **annotation** property.

By setting **coordinateUnits** value as **point** in annotation object you can placed dotted lines in the chart based on point x and y values.

INDEX.JS

```

var columnData = [{ country: "USA", gold: 50 }, { country: "China", gold:
40 }, { country: "Japan", gold: 70 },
{ country: "Australia", gold: 60 }, { country: "France", gold: 50 }, {
country: "Germany", gold: 40 },
{ country: "Italy", gold: 40 }, { country: "Sweden", gold: 30 }];

var chart = new ej.charts.Chart({
    primaryXAxis: {
        valueType: 'Category',
    },
    primaryYAxis: {

```

```

        title: 'Medals'
    },
    annotations: [{
        content: '<div id ="test" style="border-top:3px dashed
grey;border-top-width: 2px; width: 10000px"></div>',
        x: 'France',
        y: 50,
        coordinateUnits: 'Point',
        Region: 'Chart'
    }],
    series: [{
        dataSource: columnData,
        xName: 'country', yName: 'gold',
        type: 'Line'
    }],
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
    <div id="element1"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Grid data pie in ##Platform_Name## Chart control

You can visualize the filtered data that returned by grid in pie chart.

To visualize the data in pie chart, follow the given steps:

Step 1:

Initialize the grid with datasource.

Step 2:

By using the grid's `actionComplete` event and `getCurrentViewRecords` method, you can get the current page records. By setting `allowFiltering` value as `true`, you can filter the data. By using the grid's `databound` event, you can update the current page filtered records into the chart's datasource and display the grid filtered data in chart.

INDEX.JS

```

var chart = new ej.charts.Chart();
var filtertype = [
    { id: 'Menu', type: 'Menu' },
    { id: 'CheckBox', type: 'CheckBox' },
    { id: 'Excel', type: 'Excel' }
];
var grid = new ej.grids.Grid(
    {
        dataSource: orderData,
        allowPaging: true,
        allowFiltering: true,
        filterSettings: { type: 'Menu' },
        columns: [
            { field: 'OrderID', headerText: 'Order ID', width: 120,
textAlign: 'Right' },
            { field: 'CustomerName', headerText: 'Customer Name', width: 150
},
            {
                field: 'OrderDate', headerText: 'Order Date', width: 130,
                format: { type: 'dateTime', format: 'M/d/y hh:mm a' },
textAlign: 'Right'
            },
            { field: 'Freight', width: 120, format: 'C2', textAlign: 'Right'
}
        ],
        pageSettings: { pageCount: 5 },
        databound: function () {
            chart = new ej.charts.AccumulationChart({
                series: [
                    {
                        dataSource: grid.getCurrentViewRecords(),
                        type: 'Pie',
                        xName: 'CustomerName',
                        yName: 'Freight', dataLabel: { visible: true }
                    }
                ]
            }, '#Chart');
        },
        actionComplete: function (args) {
            if (args.requestType === 'paging') {
                chart.series[0].dataSource = grid.getCurrentViewRecords();
                chart.refresh();
            }
        }
    }, '#Grid');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/material.css" rel="stylesheet">
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div class="container">
    <div id="Grid"></div>
    <div id="Chart"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Down sampling in ##Platform_Name## Chart control

Downsampling is the process of reducing the data rate. We have given a 2000 data points for chart. After applying downsampling algorithm, chart data points has been reduced and rendered with 400 data points.

Downsampling data using the "Largest-Triangle-Three-Buckets algorithm"[LTTB](#) which describes the point in the bucket that forms the largest triangle using the area of the triangles. This helps to reducing the number of points.

In Downsampling when we perform zooming, particular level of zoomed chart we can see the chart clearly with original data, so we can use original data for that level of zooming. This can be achieved by [zoomComplete](#) event. Refer the below sample for downsampling with zooming feature.

INDEX.JS

```

var downdata = [
  { "x": 0, "y": 0 }, { "x": 1, "y": -5.066007678219755 }, { "x": 2, "y": -5.928724334840155 }, { "x": 3, "y": -6.1456656801181255 }, { "x": 4, "y": 4.321188375669889 }, { "x": 5, "y": 2.948801782921425 }, { "x": 6, "y": 4.348879326401061 }, { "x": 7, "y": 7.465662701191096 }, { "x": 8, "y": -7.454572839171574 }, { "x": 9, "y": -4.215304469008361 }, { "x": 10, "y": -2.576748758502486 }, { "x": 11, "y": -1.2947957433215151 }, { "x": 12, "y": -3.7712577868615975 }, { "x": 13, "y": -0.7118919161845909 }, { "x": 14,

```

```

"y": -0.9499240557155275 }, { "x": 15, "y": 0.21190635262959567 }, { "x":
16, "y": -3.287455646082684 }, { "x": 17, "y": 1.7096170465102354 }, { "x":
18, "y": -2.001941440433362 }, { "x": 19, "y": -5.290576152705585 }, { "x":
20, "y": -7.681775221827264 }, { "x": 21, "y": 8.331929996664112 }, { "x":
22, "y": 4.299712696212136 }, { "x": 23, "y": -3.8428115676594095 }, { "x":
24, "y": -4.117444700081084 }, { "x": 25, "y": -7.5428757992539515 }, { "x":
26, "y": -6.699172261203247 }, { "x": 27, "y": 5.182210187732155 }, { "x":
28, "y": -3.4590976480291733 }, { "x": 29, "y": -7.772911721385673 }, { "x":
30, "y": 7.27844392699204 }, { "x": 31, "y": 6.593711261385771 }, { "x": 32,
"y": -1.842316303386884 }, { "x": 33, "y": 4.802335350278842 }, { "x": 34,
"y": -6.166230000339985 }, { "x": 35, "y": 7.118814545886202 }, { "x": 36,
"y": -7.969517258502151 }, { "x": 37, "y": 1.9968648478110858 }, { "x": 38,
"y": 7.266963514018993 }, { "x": 39, "y": 4.111259411484399 }, { "x": 40,
"y": -5.396878273235625 }, { "x": 41, "y": 4.598609336815519 }, { "x": 42,
"y": -7.027606635376049 }, { "x": 43, "y": -5.149873665497191 }, { "x": 44,
"y": 6.185429311222256 }, { "x": 45, "y": -6.529033702766824 }, { "x": 46,
"y": -7.433349855742298 }, { "x": 47, "y": -0.5262616305652497 }, { "x": 48,
"y": 8.366066539887179 }, { "x": 49, "y": 1.371521008930003 }, { "x": 50,
"y": 4.96774643918609 }, { "x": 51, "y": -0.39091761966623295 }, { "x": 52,
"y": -3.792029934962379 }, { "x": 53, "y": -2.066317339500263 }, { "x": 54,
"y": 8.878102910778615 }, { "x": 55, "y": -0.48665593174253274 }, { "x": 56,
"y": -3.7211663290973362 }, { "x": 57, "y": 7.055857309425168 }, { "x": 58,
"y": -3.286251084568721 }, { "x": 59, "y": -4.382804726994491 }, { "x": 60,
"y": 3.7671317619194298 }, { "x": 61, "y": 3.216003691455443 }, { "x": 62,
"y": -8.654404637476492 }, { "x": 63, "y": -5.35966385984998 }, { "x": 64,
"y": -5.234620549100398 }, { "x": 65, "y": -1.5597811440277312 }, { "x": 66,
"y": 5.7649058761440966 }, { "x": 67, "y": -3.0387498549970413 }, { "x": 68,
"y": 1.5935670589970385 }, { "x": 69, "y": 8.097533639096962 }, { "x": 70,
"y": 3.570378200623672 }, { "x": 71, "y": -2.3635430074693264 }, { "x": 72,
"y": -6.738503410539572 }, { "x": 73, "y": -4.067874197034758 }, { "x": 74,
"y": 2.7636596666100566 }, { "x": 75, "y": 0.8485338327503094 }, { "x": 76,
"y": -7.650063041732215 }, { "x": 77, "y": 7.760813947187195 }, { "x": 78,
"y": -0.3426435645355177 }, { "x": 79, "y": 6.15173490733844 }, { "x": 80,
"y": 8.41998916170499 }, { "x": 81, "y": 1.3807297461555201 }, { "x": 82,
"y": -3.0223041208190775 }, { "x": 83, "y": 7.512940779748742 }, { "x": 84,
"y": 6.222676428307041 }, { "x": 85, "y": 5.978650125038886 }, { "x": 86,
"y": -8.200405540300356 }, { "x": 87, "y": -8.774508050825009 }, { "x": 88,
"y": -8.151791058146873 }, { "x": 89, "y": 6.324974668511654 }, { "x": 90,
"y": -5.207757979240489 }, { "x": 91, "y": 8.280908562994242 }, { "x": 92,
"y": 4.751016171267375 }, { "x": 93, "y": -1.7399225591310632 }, { "x": 94,
"y": -3.0451463578367184 }, { "x": 95, "y": 0.1631055461506037 }, { "x": 96,
"y": 2.519017963450935 }, { "x": 97, "y": 1.432118748085717 }, { "x": 98,
"y": -6.464700773693519 }, { "x": 99, "y": -0.5027859091400053 }, { "x":
100, "y": -8.772865445043674 }, { "x": 101, "y": 2.6465782674588976 }, {
"x": 102, "y": -7.656883387248552 }, { "x": 103, "y": 0.10090087964230854 },
{ "x": 104, "y": 4.513791535512048 }, { "x": 105, "y": -6.901857315690306 },
{ "x": 106, "y": -6.702539943544912 }, { "x": 107, "y": -3.394242278459507 },
{ "x": 108, "y": 7.66890873216456 }, { "x": 109, "y": -6.455080334650837 },
{ "x": 110, "y": 6.166403073049336 }, { "x": 111, "y": 1.3059812575354783 },
{ "x": 112, "y": 3.5807596949876856 }, { "x": 113, "y": 7.67519860721162 },
{ "x": 114, "y": -6.20090360159409 }, { "x": 115, "y": -8.648131058446584 },
{ "x": 116, "y": 2.7346140495389335 }, { "x": 117, "y": -
8.685948397830828 }, { "x": 118, "y": 0.5658992525158641 }, { "x": 119, "y":
-7.8186315275599725 }, { "x": 120, "y": -8.895025633536449 }, { "x": 121,
"y": -5.525670014613367 }, { "x": 122, "y": -7.827786409646184 }, { "x":
123, "y": -2.3917655069170554 }, { "x": 124, "y": 1.8292941022342415 }, {
"x": 125, "y": 3.6954584261222525 }, { "x": 126, "y": 5.937945085379656 }, {

```

```
"x": 127, "y": -2.9905746263837516 }, { "x": 128, "y": -5.680835694358723 },  
{ "x": 129, "y": -3.4148558602265897 }, { "x": 130, "y": -8.683720157347189  
, { "x": 131, "y": 6.460213207170002 }, { "x": 132, "y": 3.326131097368439  
, { "x": 133, "y": 0.049324651277981246 }, { "x": 134, "y":  
5.958055218017622 }, { "x": 135, "y": 2.755201470231924 }, { "x": 136, "y":  
8.800489906735809 }, { "x": 137, "y": -0.49217862333950535 }, { "x": 138,  
"y": -6.409624180803056 }, { "x": 139, "y": 2.073731129946017 }, { "x": 140,  
"y": -1.2633873031838565 }, { "x": 141, "y": -4.01705065098213 }, { "x":  
142, "y": 5.6046643578461754 }, { "x": 143, "y": -0.06420572621360066 }, {  
"x": 144, "y": 6.864262783085579 }, { "x": 145, "y": 8.39870326958508 }, {  
"x": 146, "y": 5.640367506127744 }, { "x": 147, "y": -3.609113026975125 }, {  
"x": 148, "y": -3.654649654950992 }, { "x": 149, "y": 2.7053267635191425 },  
{ "x": 150, "y": 2.8862076689901706 }, { "x": 151, "y": -7.6523957258350865  
, { "x": 152, "y": -0.619210908335166 }, { "x": 153, "y":  
1.9533755764788694 }, { "x": 154, "y": -1.9950323010288837 }, { "x": 155,  
"y": -1.8573104013560817 }, { "x": 156, "y": 8.421292902412272 }, { "x":  
157, "y": 2.1681884385199375 }, { "x": 158, "y": 7.8538824875783035 }, {  
"x": 159, "y": -8.01984897138612 }, { "x": 160, "y": -6.35257434188196 }, {  
"x": 161, "y": 3.542931180323384 }, { "x": 162, "y": 3.9550699440705497 }, {  
"x": 163, "y": -1.9412725269885103 }, { "x": 164, "y": -7.966169735023249 },  
{ "x": 165, "y": -8.406263239980472 }, { "x": 166, "y": -4.272282013909223  
, { "x": 167, "y": -7.432321007126983 }, { "x": 168, "y": 8.507279658503226  
, { "x": 169, "y": 2.9944916714381726 }, { "x": 170, "y": 2.604519295005078  
, { "x": 171, "y": 6.426937219542941 }, { "x": 172, "y": -8.13799714700996  
, { "x": 173, "y": -5.046649359167745 }, { "x": 174, "y":  
0.9352942732340672 }, { "x": 175, "y": -1.2093268097552325 }, { "x": 176,  
"y": 4.689600269136465 }, { "x": 177, "y": -2.0648056811094566 }, { "x":  
178, "y": -1.595333318813699 }, { "x": 179, "y": 0.6973453930124247 }, {  
"x": 180, "y": -2.8969537227300783 }, { "x": 181, "y": -1.9609206718765027  
, { "x": 182, "y": -1.357190533109299 }, { "x": 183, "y": -  
2.9711958714118367 }, { "x": 184, "y": 8.409072461752903 }, { "x": 185, "y":  
-0.45821387479723086 }, { "x": 186, "y": 7.685430885410618 }, { "x": 187,  
"y": 6.429873654060003 }, { "x": 188, "y": 8.624121235099611 }, { "x": 189,  
"y": -3.248006119430234 }, { "x": 190, "y": 8.856586228596939 }, { "x": 191,  
"y": -4.36999026223624 }, { "x": 192, "y": -1.8976760780244497 }, { "x":  
193, "y": -3.1584725689704767 }, { "x": 194, "y": 3.115692966843966 }, {  
"x": 195, "y": -3.425735189576777 }, { "x": 196, "y": 7.338423108726463 }, {  
"x": 197, "y": 8.591785734923608 }, { "x": 198, "y": 6.47944837861823 }, {  
"x": 199, "y": -0.012657128287459685 }, { "x": 200, "y": -3.586120032439715  
, { "x": 201, "y": 8.600816031328907 }, { "x": 202, "y": -2.505724129742185  
, { "x": 203, "y": 6.0695159316485 }, { "x": 204, "y": -5.539860824977112  
, { "x": 205, "y": -8.3259502886796 }, { "x": 206, "y": -4.95263241713916  
, { "x": 207, "y": -4.372405290093487 }, { "x": 208, "y": -  
6.209984690521315 }, { "x": 209, "y": -8.028082421929504 }, { "x": 210, "y":  
1.3373080540763898 }, { "x": 211, "y": 4.356669481966348 }, { "x": 212, "y":  
-1.4702984196916002 }, { "x": 213, "y": -5.696889705079843 }, { "x": 214,  
"y": -8.6874341232908 }, { "x": 215, "y": -7.372729022914079 }, { "x": 216,  
"y": 2.6539443565140743 }, { "x": 217, "y": 2.1996606852985874 }, { "x":  
218, "y": 6.109690022333542 }, { "x": 219, "y": -8.153880489258666 }, { "x":  
220, "y": -6.210899614276704 }, { "x": 221, "y": 1.7615798191920646 }, {  
"x": 222, "y": -0.7141568297794798 }, { "x": 223, "y": -3.0776981116374964  
, { "x": 224, "y": -1.315013481600781 }, { "x": 225, "y": 6.568497900882626  
, { "x": 226, "y": -8.336949941165216 }, { "x": 227, "y": 8.365496928690941  
, { "x": 228, "y": -4.033493635345434 }, { "x": 229, "y": -  
7.009428250861919 }, { "x": 230, "y": -3.7296243748502427 }, { "x": 231,  
"y": 1.6778563974746596 }, { "x": 232, "y": -1.1574553775805336 }, { "x":  
233, "y": 8.849047469006386 }, { "x": 234, "y": 0.10911457504197308 }, {
```

```
"x": 235, "y": 2.6171318636524745 }, { "x": 236, "y": -7.614252344684505 },  
{ "x": 237, "y": 0.9297034976780783 }, { "x": 238, "y": -5.769921467248013  
, { "x": 239, "y": -8.854142254467348 }, { "x": 240, "y": -  
3.4368359065464693 }, { "x": 241, "y": 3.651033630509774 }, { "x": 242, "y":  
0.717049878824092 }, { "x": 243, "y": 7.772803204099461 }, { "x": 244, "y":  
-7.856242868211638 }, { "x": 245, "y": -8.40104564218717 }, { "x": 246, "y":  
5.707525753710122 }, { "x": 247, "y": 1.9618562057163 }, { "x": 248, "y":  
7.004935730942364 }, { "x": 249, "y": 4.4960513165682165 }, { "x": 250, "y":  
-4.825893968850941 }, { "x": 251, "y": -3.2873822430231288 }, { "x": 252,  
"y": -8.299779534046813 }, { "x": 253, "y": -2.15375542062419 }, { "x": 254,  
"y": -0.7116327214904068 }, { "x": 255, "y": 4.628226400220834 }, { "x":  
256, "y": -2.525349340056442 }, { "x": 257, "y": 4.747856359708042 }, { "x":  
258, "y": -4.259159845238977 }, { "x": 259, "y": -0.6267185882764625 },  
{ "x": 260, "y": 8.931905782820671 }, { "x": 261, "y": -  
5.6467545282637985 }, { "x": 262, "y": 7.3790364215473225 }, { "x": 263,  
"y": 8.863780740568377 }, { "x": 264, "y": 1.2161876606997204 }, { "x": 265,  
"y": 3.938293593316674 }, { "x": 266, "y": 1.8179619807790957 }, { "x": 267,  
"y": -2.1248883594678816 }, { "x": 268, "y": -3.27089239229665 }, { "x":  
269, "y": -4.123695815816533 }, { "x": 270, "y": -0.6782096714278048 }, {  
"x": 271, "y": 6.719617919183941 }, { "x": 272, "y": -6.296018826096353 }, {  
"x": 273, "y": -3.609738976046489 }, { "x": 274, "y": -2.6591995427709687 },  
{ "x": 275, "y": 0.671306751109233 }, { "x": 276, "y": -1.819546845130592 },  
{ "x": 277, "y": -2.0014653383121104 }, { "x": 278, "y": 4.584789493815725  
, { "x": 279, "y": -5.035367621264342 }, { "x": 280, "y":  
3.0412888255048713 }, { "x": 281, "y": -7.308076407041668 }, { "x": 282,  
"y": -3.466628569200582 }, { "x": 283, "y": -5.067496676185206 }, { "x":  
284, "y": -1.0669654999372575 }, { "x": 285, "y": -8.87561786285942 }, {  
"x": 286, "y": -5.497534447328873 }, { "x": 287, "y": 3.6581908806178234 },  
{ "x": 288, "y": -7.2507606680012495 }, { "x": 289, "y": -3.2361495736467205  
, { "x": 290, "y": -8.05695190283468 }, { "x": 291, "y": 5.355211755749066  
, { "x": 292, "y": 6.495480066314613 }, { "x": 293, "y": -5.557060444037582  
, { "x": 294, "y": 1.5389452624371955 }, { "x": 295, "y": 8.507375550676379  
, { "x": 296, "y": 1.7820724457800097 }, { "x": 297, "y":  
1.5839580746007087 }, { "x": 298, "y": 7.00766570878168 }, { "x": 299, "y":  
-7.650805094249352 }, { "x": 300, "y": 1.187528465202865 }, { "x": 301, "y":  
-5.678076248292863 }, { "x": 302, "y": 1.7819569957275654 }, { "x": 303,  
"y": 2.3249089301398183 }, { "x": 304, "y": 4.689057069539706 }, { "x": 305,  
"y": 0.015672445677747504 }, { "x": 306, "y": 7.348142618599667 }, { "x":  
307, "y": -7.333216317522609 }, { "x": 308, "y": -2.1833554226761116 }, {  
"x": 309, "y": 5.383274460303301 }, { "x": 310, "y": -3.613163255212733 }, {  
"x": 311, "y": 8.414129307049993 }, { "x": 312, "y": 2.2479553923696294 }, {  
"x": 313, "y": 2.162622134375983 }, { "x": 314, "y": -1.8167901937237287 },  
{ "x": 315, "y": 0.47544084446782975 }, { "x": 316, "y": -2.946877137870768  
, { "x": 317, "y": -2.962854939209679 }, { "x": 318, "y": 8.86681435662825  
, { "x": 319, "y": 2.0462727377095753 }, { "x": 320, "y":  
6.0956390828083915 }, { "x": 321, "y": -6.060832702896317 }, { "x": 322,  
"y": 6.24172781906643 }, { "x": 323, "y": 0.7604094395572893 }, { "x": 324,  
"y": 4.8444448401193507 }, { "x": 325, "y": 4.857732746894397 }, { "x": 326,  
"y": -6.21879998366952 }, { "x": 327, "y": -7.959122822606876 }, { "x": 328,  
"y": 3.741339564474462 }, { "x": 329, "y": -5.113229048690366 }, { "x": 330,  
"y": -1.2475964298410958 }, { "x": 331, "y": -8.450777213013653 }, { "x":  
332, "y": -4.200783107160451 }, { "x": 333, "y": 8.844266102151103 }, { "x":  
334, "y": 5.245122920351562 }, { "x": 335, "y": -8.719043528523292 }, { "x":  
336, "y": -7.983297348304871 }, { "x": 337, "y": -2.1451955245090666 }, {  
"x": 338, "y": -7.741225370758689 }, { "x": 339, "y": 7.721750624063215 }, {  
"x": 340, "y": 8.77813196959228 }, { "x": 341, "y": 7.114735265567159 }, {  
"x": 342, "y": -0.9591870819557222 }, { "x": 343, "y": -1.3711924968194378
```



```

}, { "x": 344, "y": -0.6258185625450956 }, { "x": 345, "y":
2.322736633819371 }, { "x": 346, "y": 6.76450584019071 }, { "x": 347, "y":
4.767177010391244 }, { "x": 348, "y": 4.686851005836763 }, { "x": 349, "y":
2.5566486855993737 }, { "x": 350, "y": -3.1586311113576473 }, { "x": 351,
"y": 5.927722985919008 }, { "x": 352, "y": 5.286870643348669 }, { "x": 353,
"y": 3.117483687054598 }, { "x": 354, "y": 1.649534807246532 }, { "x": 355,
"y": -1.4280653398736822 }, { "x": 356, "y": -1.1459757842429656 }, { "x":
357, "y": 2.8995080210724886 }, { "x": 358, "y": 2.014178793403895 }, { "x":
359, "y": 6.06680250387215 }, { "x": 360, "y": -5.827415787116241 }, { "x":
361, "y": 4.6551265758355616 }, { "x": 362, "y": 3.9795406062916534 }, {
"x": 363, "y": 4.1315433419531065 }, { "x": 364, "y": 1.3212015825369754 },
{ "x": 365, "y": -4.1936328868129475 }, { "x": 366, "y": -8.47438827046254
}, { "x": 367, "y": 8.645516522773452 }, { "x": 368, "y": -2.082255453454656
}, { "x": 369, "y": 7.089594452066308 }, { "x": 370, "y": -3.996386812449087
}, { "x": 371, "y": -8.615417038372291 }, { "x": 372, "y": 3.372641648042787
}, { "x": 373, "y": -0.019531131754423114 }, { "x": 374, "y":
6.7191375622142395 }, { "x": 375, "y": -0.0730872556065787 }, { "x": 376,
"y": -2.9642144198521674 }, { "x": 377, "y": -3.896896957188523 }, { "x":
378, "y": 2.737084292758114 }, { "x": 379, "y": -6.121211294972079 }, { "x":
380, "y": 4.63077015966619 }, { "x": 381, "y": -0.9960059218675426 }, { "x":
382, "y": -5.290802081048746 }, { "x": 383, "y": 0.652321382325427 }, { "x":
384, "y": -6.379141480833464 }, { "x": 385, "y": 3.9993376569034087 }, {
"x": 386, "y": 0.7076171767818842 }, { "x": 387, "y": -6.319651189792667 },
{ "x": 388, "y": 7.3457792775877095 }, { "x": 389, "y": -5.627238290316177
}, { "x": 390, "y": -7.538825171458681 }, { "x": 391, "y": 7.569815715193172
}, { "x": 392, "y": -2.1927560291044843 }, { "x": 393, "y":
1.3847989729317085 }, { "x": 394, "y": -2.4391586485844847 }, { "x": 395,
"y": 2.2798850267506 }, { "x": 396, "y": -7.842015422664817 }, { "x": 397,
"y": -1.2981272241798054 }, { "x": 398, "y": 5.25972329195112 }, { "x": 399,
"y": -4.7392946529708135 }, { "x": 400, "y": -6.583600412678105 }, { "x":
401, "y": -0.2842618090194975 }, { "x": 402, "y": 7.454727248370048 }, {
"x": 403, "y": -0.6707799102294469 }, { "x": 404, "y": -1.2734657543022845
}, { "x": 405, "y": 4.351909125244934 }, { "x": 406, "y": -8.882043273185813
}, { "x": 407, "y": -7.9237632050499816 }, { "x": 408, "y": -
6.401846771517016 }, { "x": 409, "y": 2.689353460056818 }, { "x": 410, "y":
3.36484446173934 }, { "x": 411, "y": -1.9946356448911686 }, { "x": 412, "y":
-0.25477833958279916 }, { "x": 413, "y": -8.86739133368448 }, { "x": 414,
"y": 5.022894995604846 }, { "x": 415, "y": 2.1396439373406615 }, { "x": 416,
"y": 8.849147509506217 }, { "x": 417, "y": 2.79911635102917 }, { "x": 418,
"y": -2.1849934880112576 }, { "x": 419, "y": 0.2992471144044444 }, { "x":
420, "y": 7.679306110780757 }, { "x": 421, "y": 7.101514333716882 }, { "x":
422, "y": 3.9488676198920167 }, { "x": 423, "y": -7.128435499448014 }, {
"x": 424, "y": -4.534348123804083 }, { "x": 425, "y": 6.643919667612931 }, {
"x": 426, "y": 0.7453803481603707 }, { "x": 427, "y": -0.10580956987317158
}, { "x": 428, "y": 2.7648238999440267 }, { "x": 429, "y": -
2.3830321422340823 }, { "x": 430, "y": 7.482146661848262 }, { "x": 431, "y":
5.6136220328445265 }, { "x": 432, "y": 5.14871890911631 }, { "x": 433, "y":
4.796042406639616 }, { "x": 434, "y": 3.3956224094151537 }, { "x": 435, "y":
6.756711057952909 }, { "x": 436, "y": 5.941394483269981 }, { "x": 437, "y":
8.516820005612907 }, { "x": 438, "y": 1.1050500219430326 }, { "x": 439, "y":
7.372123614668407 }, { "x": 440, "y": 2.3075854831782614 }, { "x": 441, "y":
-7.590213004353343 }, { "x": 442, "y": 2.3068393959653832 }, { "x": 443,
"y": -0.8162224834305203 }, { "x": 444, "y": 0.9942098100033512 }, { "x":
445, "y": 1.074429539045946 }, { "x": 446, "y": -0.6203446010345726 }, {
"x": 447, "y": -7.085563900098249 }, { "x": 448, "y": -7.885592816171497 },
{ "x": 449, "y": 5.911340520440676 }, { "x": 450, "y": 8.251104937032565 },
{ "x": 451, "y": 2.7063650497323195 }, { "x": 452, "y": -5.260746577399921

```

```

}, { "x": 453, "y": 1.7838235743990651 }, { "x": 454, "y": 6.13380320900583
}, { "x": 455, "y": 3.68637008456205 }, { "x": 456, "y": -4.280584302930965
}, { "x": 457, "y": 5.579111093229182 }, { "x": 458, "y": -8.239410225936036
}, { "x": 459, "y": -3.777147756651038 }, { "x": 460, "y": 8.498074407326541
}, { "x": 461, "y": 5.879181959877579 }, { "x": 462, "y": 3.915657405269455
}, { "x": 463, "y": -1.901081078023143 }, { "x": 464, "y":
3.8926800535396424 }, { "x": 465, "y": 2.608213227513014 }, { "x": 466, "y":
7.637401332766235 }, { "x": 467, "y": 8.222088017223072 }, { "x": 468, "y":
-1.0878408694149924 }, { "x": 469, "y": 6.768093948675521 }, { "x": 470,
"y": -1.4887011547155398 }, { "x": 471, "y": 7.021285939584551 }, { "x":
472, "y": -8.560994139952946 }, { "x": 473, "y": -2.2344313368410598 }, {
"x": 474, "y": 2.170007884189925 }, { "x": 475, "y": 2.642171911160913 }, {
"x": 476, "y": -5.601304199651059 }, { "x": 477, "y": 8.203292471251483 }, {
"x": 478, "y": -6.445129657231408 }, { "x": 479, "y": 4.870580616972491 }, {
"x": 480, "y": 5.309057592505397 }, { "x": 481, "y": -1.2262515102831468 },
{ "x": 482, "y": -3.31518896988071 }, { "x": 483, "y": 0.35542859706493424
}, { "x": 484, "y": -0.4662287078572085 }, { "x": 485, "y":
4.731146864495903 }, { "x": 486, "y": 5.463291694207079 }, { "x": 487, "y":
0.9752286935873204 }, { "x": 488, "y": -7.926999858903878 }, { "x": 489,
"y": -4.409532812892482 }, { "x": 490, "y": 4.667740091404703 }, { "x": 491,
"y": 7.403871951278763 }, { "x": 492, "y": 0.14458535810906525 }, { "x":
493, "y": -2.4129748509189195 }, { "x": 494, "y": -6.905740300502895 }, {
"x": 495, "y": 8.287913653542748 }, { "x": 496, "y": 3.4073353069456065 }, {
"x": 497, "y": -3.855937137662366 }, { "x": 498, "y": 1.76192888891892 }, {
"x": 499, "y": -8.481861931064895 }, { "x": 500, "y": -5.623623101956982 },
{ "x": 501, "y": 8.62935477210381 }, { "x": 502, "y": -1.1736238398817118 },
{ "x": 503, "y": 6.085284798867722 }, { "x": 504, "y": 5.702448054918364 },
{ "x": 505, "y": 0.27070680881163334 }, { "x": 506, "y": -4.094898269425724
}, { "x": 507, "y": -0.1309095677178007 }, { "x": 508, "y": -
3.3813118745532176 }, { "x": 509, "y": 1.5999862844036716 }, { "x": 510,
"y": -1.1769989936167597 }, { "x": 511, "y": -1.9453662517040051 }, { "x":
512, "y": 3.61276113628708 }, { "x": 513, "y": -1.1048492729761588 }, { "x":
514, "y": 3.3413316729944746 }, { "x": 515, "y": 1.5870036075951468 }, {
"x": 516, "y": -1.4946527621223105 },
{ "x": 517, "y": 7.848426983965378 }, { "x": 518, "y": 5.924045529628511
}, { "x": 519, "y": 1.847408218682899 }, { "x": 520, "y": 3.6195573596150012
}, { "x": 521, "y": 2.575286621681407 }, { "x": 522, "y": -
0.18749839156613035 }, { "x": 523, "y": -6.067762029839244 }, { "x": 524,
"y": 4.9538910753438135 }, { "x": 525, "y": 5.692429118583286 }, { "x": 526,
"y": -0.5228345705073636 }, { "x": 527, "y": -2.7819551375036067 }, { "x":
528, "y": 3.0906676979786063 }, { "x": 529, "y": 4.195357450969304 }, { "x":
530, "y": 0.6790143043311083 }, { "x": 531, "y": 0.6215337677450279 }, {
"x": 532, "y": -2.7615394219687115 }, { "x": 533, "y": 3.902964747830202 },
{ "x": 534, "y": -8.368876421978399 }, { "x": 535, "y": -7.184928450698217
}, { "x": 536, "y": -0.3393479043576413 }, { "x": 537, "y": -
4.932058047162194 }, { "x": 538, "y": 3.005689660641945 }, { "x": 539, "y":
-1.736354927027782 }, { "x": 540, "y": 0.3059376691846367 }, { "x": 541,
"y": -5.201779963744382 }, { "x": 542, "y": -5.173299873480324 }, { "x":
543, "y": -0.7652246643475564 }, { "x": 544, "y": -3.1609381248283146 }, {
"x": 545, "y": -0.6012518736227861 }, { "x": 546, "y": -7.731179496258861 },
{ "x": 547, "y": -1.7758497875163366 }, { "x": 548, "y": -0.3154877341177418
}, { "x": 549, "y": 4.75892515777875 }, { "x": 550, "y": -6.243558980462517
}, { "x": 551, "y": 1.2849357553891334 }, { "x": 552, "y": -
8.407541358960035 }, { "x": 553, "y": 6.75442774228398 }, { "x": 554, "y":
8.823862728787311 }, { "x": 555, "y": -3.358069248573784 }, { "x": 556, "y":
7.927939006723115 }, { "x": 557, "y": -6.3888962968496426 }, { "x": 558,
"y": -3.1658101678108856 }, { "x": 559, "y": 4.134107639637156 }, { "x":

```

```
560, "y": 7.076201782675135 }, { "x": 561, "y": -7.393497811413015 }, { "x":  
562, "y": 7.00409761178679 }, { "x": 563, "y": -1.6565223868491392 }, { "x":  
564, "y": 1.0908522977194473 }, { "x": 565, "y": 4.436780005929551 }, { "x":  
566, "y": -2.5513914184878637 }, { "x": 567, "y": 3.1274970914400093 }, {  
"x": 568, "y": 6.45074094833922 }, { "x": 569, "y": -7.683849914874396 }, {  
"x": 570, "y": 4.785878529663993 }, { "x": 571, "y": -7.890679444713726 }, {  
"x": 572, "y": -0.9868260517998984 }, { "x": 573, "y": -0.9996205421353661  
, { "x": 574, "y": -5.655136900872027 }, { "x": 575, "y": -5.26091604754198  
, { "x": 576, "y": 4.184129397558129 }, { "x": 577, "y": -5.759848180744694  
, { "x": 578, "y": -4.186603581196085 }, { "x": 579, "y": 4.644233446414464  
, { "x": 580, "y": -5.421238577417596 }, { "x": 581, "y": 1.586578989606533  
, { "x": 582, "y": -4.120624324706053 }, { "x": 583, "y": -  
3.6126488197915787 }, { "x": 584, "y": -1.756730103194755 }, { "x": 585,  
"y": -4.554937904701989 }, { "x": 586, "y": 5.345545103570402 }, { "x": 587,  
"y": 0.5933292706425721 }, { "x": 588, "y": -0.8148504038213868 }, { "x":  
589, "y": -6.24547893844066 }, { "x": 590, "y": 7.972305464926578 }, { "x":  
591, "y": -1.0162103419652109 }, { "x": 592, "y": -6.776187753521496 }, {  
"x": 593, "y": -3.9624592160439436 }, { "x": 594, "y": -8.790915680218717 },  
{ "x": 595, "y": -7.2462507979028254 }, { "x": 596, "y": 6.72363446925096 },  
{ "x": 597, "y": -2.5994476088742138 }, { "x": 598, "y": 2.026194539086827  
, { "x": 599, "y": 7.862039901517548 }, { "x": 600, "y": 5.39046281300673  
, { "x": 601, "y": -0.04703229887467053 }, { "x": 602, "y": -  
6.323295767040616 }, { "x": 603, "y": -8.272350329913957 }, { "x": 604, "y":  
-0.5379452811352081 }, { "x": 605, "y": 6.655074851449747 }, { "x": 606,  
"y": -3.986477390590732 }, { "x": 607, "y": -6.46644385437277 }, { "x": 608,  
"y": 7.398447720703928 }, { "x": 609, "y": -2.581938197645103 }, { "x": 610,  
"y": 6.744752371844726 }, { "x": 611, "y": 3.168820015674182 }, { "x": 612,  
"y": -8.61912746325466 }, { "x": 613, "y": 7.3120114316180995 }, { "x": 614,  
"y": -8.398693983532478 }, { "x": 615, "y": -0.18768044990050825 }, { "x":  
616, "y": -2.312460779272288 }, { "x": 617, "y": 7.149571425390377 }, { "x":  
618, "y": 6.118387903469985 }, { "x": 619, "y": 1.6556968226669433 }, { "x":  
620, "y": -0.05772275028525442 }, { "x": 621, "y": -8.268903023656911 }, {  
"x": 622, "y": 1.672698250575758 }, { "x": 623, "y": -4.772154884399737 }, {  
"x": 624, "y": 6.953951494287638 }, { "x": 625, "y": 8.194374200620864 }, {  
"x": 626, "y": -4.716481805998422 }, { "x": 627, "y": 3.6053427051765965 },  
{ "x": 628, "y": 4.857650667967306 }, { "x": 629, "y": 2.5978327763505664 },  
{ "x": 630, "y": 4.594758427385242 }, { "x": 631, "y": -0.8847210884053229  
, { "x": 632, "y": 4.197790225509838 }, { "x": 633, "y": -4.414005077557183  
, { "x": 634, "y": -0.6225738570349577 }, { "x": 635, "y":  
3.674000879065785 }, { "x": 636, "y": 4.302045450199774 }, { "x": 637, "y":  
-2.971193727685792 }, { "x": 638, "y": 4.990353258017283 }, { "x": 639, "y":  
-6.13322768398092 }, { "x": 640, "y": 3.650997948204356 }, { "x": 641, "y":  
0.01542674068549843 }, { "x": 642, "y": -8.035846533722403 }, { "x": 643,  
"y": 3.1654103145775654 }, { "x": 644, "y": 5.739402526659756 }, { "x": 645,  
"y": 1.543523121699586 }, { "x": 646, "y": -7.230621031234257 }, { "x": 647,  
"y": 6.000108754195157 }, { "x": 648, "y": -7.408991171181964 }, { "x": 649,  
"y": -1.020531826817484 }, { "x": 650, "y": -8.444430021532257 }, { "x":  
651, "y": -3.8732530604547284 }, { "x": 652, "y": 6.907777569014087 }, {  
"x": 653, "y": -1.4732199155436172 }, { "x": 654, "y": 2.0776714633211952 },  
{ "x": 655, "y": 5.167636692459585 }, { "x": 656, "y": -6.635558813891742 },  
{ "x": 657, "y": -2.6244477573472116 }, { "x": 658, "y": 5.355707750092947  
, { "x": 659, "y": 0.8928561023305903 }, { "x": 660, "y": 5.713200837935155  
, { "x": 661, "y": -4.224260077950406 }, { "x": 662, "y": -  
8.816246015992487 }, { "x": 663, "y": -2.478184927753313 }, { "x": 664, "y":  
-5.030761100656802 }, { "x": 665, "y": -1.4041556582144752 }, { "x": 666,  
"y": 0.14624441544577493 }, { "x": 667, "y": 1.2322035993184617 }, { "x":  
668, "y": -8.075283508412745 }, { "x": 669, "y": -5.63022129453939 }, { "x":
```

```

670, "y": 8.375559930192285 }, { "x": 671, "y": -6.5798621630296275 }, {
"x": 672, "y": -0.31799909608572463 }, { "x": 673, "y": 8.84994788020704 },
{ "x": 674, "y": 3.2582427118725885 }, { "x": 675, "y": 2.903748706922485 },
{ "x": 676, "y": 7.191027861418288 }, { "x": 677, "y": -7.207429637956208 },
{ "x": 678, "y": 0.6992571774160705 }, { "x": 679, "y": -6.662022053920833
}, { "x": 680, "y": -5.919898123026888 }, { "x": 681, "y": 3.583679553385979
}, { "x": 682, "y": -8.477684601238625 }, { "x": 683, "y": -
6.354608997514381 }, { "x": 684, "y": -7.868459274916017 }, { "x": 685, "y":
-7.254171229679647 }, { "x": 686, "y": 4.11010915829597 }, { "x": 687, "y":
7.5613709860386855 }, { "x": 688, "y": 2.919498619021578 }, { "x": 689, "y":
1.1672926911017942 }, { "x": 690, "y": 1.157812346482599 }, { "x": 691, "y":
1.1347129561340452 }, { "x": 692, "y": -3.1657634003893858 }, { "x": 693,
"y": -2.7794023903913683 }, { "x": 694, "y": -6.315288645973235 }, { "x":
695, "y": 7.129924406047582 }, { "x": 696, "y": -5.658235949193164 }, { "x":
697, "y": -1.4839747844539586 }, { "x": 698, "y": 5.616428820983176 }, {
"x": 699, "y": -7.273853143842013 }, { "x": 700, "y": 8.157018481657904 }, {
"x": 701, "y": -0.9853521501410931 }, { "x": 702, "y": -3.6491243605287558
}, { "x": 703, "y": -6.454946062910491 }, { "x": 704, "y": -
4.832144753225778 }, { "x": 705, "y": -7.611831301191797 }, { "x": 706, "y":
-2.3154370348029776 }, { "x": 707, "y": 7.54373357360388 }, { "x": 708, "y":
1.3188553707704465 }, { "x": 709, "y": 2.930207805043583 }, { "x": 710, "y":
-4.150779608246797 }, { "x": 711, "y": 0.36074082210786784 }, { "x": 712,
"y": -0.6391538519402538 }, { "x": 713, "y": 6.990263839940317 }, { "x":
714, "y": -8.984285291670298 }, { "x": 715, "y": 4.664142885446038 }, { "x":
716, "y": -4.6948227937926665 }, { "x": 717, "y": -1.1072533049766395 }, {
"x": 718, "y": 4.8247565860226835 }, { "x": 719, "y": 3.6628139084913407 },
{ "x": 720, "y": -2.4529269689476774 }, { "x": 721, "y": 0.392481045391186
}, { "x": 722, "y": -1.4381667868357866 }, { "x": 723, "y":
0.7298706848499901 }, { "x": 724, "y": 0.1309355217145125 }, { "x": 725,
"y": -3.346861784604112 }, { "x": 726, "y": -8.18788750850278 }, { "x": 727,
"y": -1.0422009220088224 }, { "x": 728, "y": -5.515262688864823 }, { "x":
729, "y": -5.6241386032708895 }, { "x": 730, "y": 8.579794369523675 }, {
"x": 731, "y": -5.271212347347553 }, { "x": 732, "y": 3.457389935219016 }, {
"x": 733, "y": -4.9956084160897625 }, { "x": 734, "y": 7.350989442277786 },
{ "x": 735, "y": -8.173184698759453 }, { "x": 736, "y": -6.575187017419355
}, { "x": 737, "y": -4.597943350380731 }, { "x": 738, "y": -
2.7232612494707933 }, { "x": 739, "y": -7.121042869416945 }, { "x": 740,
"y": 7.810194988009432 }, { "x": 741, "y": 7.946563521375612 }, { "x": 742,
"y": 1.6686784839680655 }, { "x": 743, "y": 6.6713591770447 }, { "x": 744,
"y": 8.161510862033815 }, { "x": 745, "y": -0.0402131056495012 }, { "x":
746, "y": 5.319035596798024 }, { "x": 747, "y": 8.484722539205421 }, { "x":
748, "y": -5.273454599540388 }, { "x": 749, "y": -5.218969398978725 }, {
"x": 750, "y": 1.3952219880942565 }, { "x": 751, "y": -7.7985576547063 }, {
"x": 752, "y": 0.27623106032860356 }, { "x": 753, "y": 6.271749893268884 },
{ "x": 754, "y": -7.725754843316649 }, { "x": 755, "y": -2.489459911934027
}, { "x": 756, "y": 8.265299457466543 }, { "x": 757, "y": -7.964447046482112
}, { "x": 758, "y": -3.433562914793171 }, { "x": 759, "y": -
1.969886968120072 }, { "x": 760, "y": 7.648726166117903 }, { "x": 761, "y":
6.152657775467748 }, { "x": 762, "y": -6.589138059852487 }, { "x": 763, "y":
8.03435975680977 }, { "x": 764, "y": -6.455686668667125 }, { "x": 765, "y":
8.234987079287567 }, { "x": 766, "y": -4.479026051799233 }, { "x": 767, "y":
-0.4650654367261744 }, { "x": 768, "y": 1.7227670393309182 }, { "x": 769,
"y": 2.943265239015794 }, { "x": 770, "y": 5.580769371445774 }, { "x": 771,
"y": 4.265417973994511 }, { "x": 772, "y": -1.0179648194202011 }, { "x":
773, "y": -7.017680561139949 },
{ "x": 774, "y": 2.843083132733746 }, { "x": 775, "y":
1.8475274135080895 }, { "x": 776, "y": 8.577564517460317 }, { "x": 777, "y":

```

```
2.6277983170441903 }, { "x": 778, "y": -6.909230401684853 }, { "x": 779,
"y": 2.5568492437747814 }, { "x": 780, "y": -4.257296607687834 }, { "x":
781, "y": 2.878937913887672 }, { "x": 782, "y": -0.7764895901290814 }, {
"x": 783, "y": -2.3720737504846667 }, { "x": 784, "y": -1.5036436901610353
}, { "x": 785, "y": -6.5507031023490665 }, { "x": 786, "y":
7.627255385487615 }, { "x": 787, "y": -4.959449597907609 }, { "x": 788, "y":
-7.816669951473974 }, { "x": 789, "y": 1.7929187340270438 }, { "x": 790,
"y": 2.1835715284882973 }, { "x": 791, "y": 7.198967189229762 }, { "x": 792,
"y": -0.9317649803817432 }, { "x": 793, "y": -3.962185279148656 }, { "x":
794, "y": -1.2799622720178494 }, { "x": 795, "y": -1.459724117227573 }, {
"x": 796, "y": 0.28356852658343357 }, { "x": 797, "y": -8.313575716069728 },
{ "x": 798, "y": 2.9216447532872873 }, { "x": 799, "y": 0.8215888840348189
}, { "x": 800, "y": 8.707409301399071 }, { "x": 801, "y": 4.362455906383705
}, { "x": 802, "y": 8.028947464503936 }, { "x": 803, "y": 0.0815043949284231
}, { "x": 804, "y": -0.8263005610641336 }, { "x": 805, "y":
2.4797136866677487 }, { "x": 806, "y": -8.54450551321274 }, { "x": 807, "y":
0.7824415257916826 }, { "x": 808, "y": -5.83694465564398 }, { "x": 809, "y":
-5.306217150792795 }, { "x": 810, "y": 6.991883712962627 }, { "x": 811, "y":
-2.857463473666761 }, { "x": 812, "y": -4.511545200120098 }, { "x": 813,
"y": -0.18498211674415188 }, { "x": 814, "y": -1.2710212072145453 }, { "x":
815, "y": -1.7220201038704541 }, { "x": 816, "y": -0.9771170056501628 }, {
"x": 817, "y": 2.584715422416094 }, { "x": 818, "y": 3.175378169393257 }, {
"x": 819, "y": -4.553377645527005 }, { "x": 820, "y": 2.6257092816414556 },
{ "x": 821, "y": -5.081319067288543 }, { "x": 822, "y": -8.01036226283945 },
{ "x": 823, "y": 6.106754641065786 }, { "x": 824, "y": -0.9959542881656365
}, { "x": 825, "y": 8.114403955322835 }, { "x": 826, "y": -3.421739521294839
}, { "x": 827, "y": -0.06952336735894704 }, { "x": 828, "y": -
4.641154890002444 }, { "x": 829, "y": 5.946389982378943 }, { "x": 830, "y":
-8.326016924551908 }, { "x": 831, "y": 8.862643297232488 }, { "x": 832, "y":
-1.3336123383831868 }, { "x": 833, "y": 3.7798052469835426 }, { "x": 834,
"y": 6.750917425529213 }, { "x": 835, "y": 3.9684182063352296 }, { "x": 836,
"y": 1.589583939105971 }, { "x": 837, "y": 4.725618687294187 }, { "x": 838,
"y": 5.461508832957037 }, { "x": 839, "y": -6.809280962447031 }, { "x": 840,
"y": 1.3317022957769389 }, { "x": 841, "y": -6.072082094488762 }, { "x":
842, "y": 6.825261395885201 }, { "x": 843, "y": -3.345462233015839 }, { "x":
844, "y": -7.491633709189054 }, { "x": 845, "y": 8.1564317288545 }, { "x":
846, "y": -8.01219620231602 }, { "x": 847, "y": 5.154420805462488 }, { "x":
848, "y": -5.829137745145463 }, { "x": 849, "y": -1.1823122731743965 }, {
"x": 850, "y": -8.631871581619414 }, { "x": 851, "y": 5.119269782479254 }, {
"x": 852, "y": -7.963107856240548 }, { "x": 853, "y": 4.8020682521584455 },
{ "x": 854, "y": 7.117302103187839 }, { "x": 855, "y": -2.1470155118455363
}, { "x": 856, "y": 6.532104520083134 }, { "x": 857, "y": 3.972744268899657
}, { "x": 858, "y": 1.9440758207638016 }, { "x": 859, "y": -
2.4160834674902034 }, { "x": 860, "y": -4.768220950391724 }, { "x": 861,
"y": 6.3594397566581655 }, { "x": 862, "y": 4.623802517187389 }, { "x": 863,
"y": 6.48294592240717 }, { "x": 864, "y": 6.17291906112151 }, { "x": 865,
"y": 6.604709750769931 }, { "x": 866, "y": -0.7006520353445627 }, { "x":
867, "y": -1.1761641709493533 }, { "x": 868, "y": -0.08522900596513239 }, {
"x": 869, "y": -5.233899291474326 }, { "x": 870, "y": 6.36654832980145 }, {
"x": 871, "y": 6.577328390474044 }, { "x": 872, "y": -5.1774933968659544 },
{ "x": 873, "y": -8.370631698238851 }, { "x": 874, "y": -0.40950193199079976
}, { "x": 875, "y": -2.9482902779175513 }, { "x": 876, "y":
0.09879042972414709 }, { "x": 877, "y": 2.1323739460965907 }, { "x": 878,
"y": 0.401407821615539 }, { "x": 879, "y": 1.618077028621693 }, { "x": 880,
"y": -4.956297035460082 }, { "x": 881, "y": -0.3424218868438107 }, { "x":
882, "y": -0.6506176395192025 }, { "x": 883, "y": 5.058907536617802 }, {
"x": 884, "y": 7.31757334281847 }, { "x": 885, "y": 8.504604235042269 }, {
```

```
"x": 886, "y": 6.192046419412048 }, { "x": 887, "y": -0.9944057806954216 },  
{ "x": 888, "y": 3.1334767790513887 }, { "x": 889, "y": 4.850494123489803 },  
{ "x": 890, "y": 0.7688152702791022 }, { "x": 891, "y": 4.563249832601851 },  
{ "x": 892, "y": 4.400037193233118 }, { "x": 893, "y": 1.8459968620984082 },  
{ "x": 894, "y": 3.392871194711507 }, { "x": 895, "y": 4.75246434501703 }, {  
"x": 896, "y": -1.196812146746483 }, { "x": 897, "y": -1.6350377780472698 },  
{ "x": 898, "y": 0.4985752008749511 }, { "x": 899, "y": -3.3103442849899825  
}, { "x": 900, "y": -2.8104517312422885 }, { "x": 901, "y":  
6.4746762484053715 }, { "x": 902, "y": -8.755269822929769 }, { "x": 903,  
"y": 4.5194237459607844 }, { "x": 904, "y": 1.4444282254455878 }, { "x":  
905, "y": 6.918051581948355 }, { "x": 906, "y": 1.9114514254162565 }, { "x":  
907, "y": 4.63126885622005 }, { "x": 908, "y": -6.444083344166446 }, { "x":  
909, "y": -1.3804331192257164 }, { "x": 910, "y": -7.2550034900218945 }, {  
"x": 911, "y": -6.011878886653854 }, { "x": 912, "y": -8.05266684427896 }, {  
"x": 913, "y": 6.632946196884863 }, { "x": 914, "y": 0.507815827354996 }, {  
"x": 915, "y": -0.6510747239612158 }, { "x": 916, "y": -2.3580057177938025  
}, { "x": 917, "y": 1.7348349575152398 }, { "x": 918, "y": -  
5.119493840964892 }, { "x": 919, "y": 1.0134654507357794 }, { "x": 920, "y":  
2.8592263772853137 }, { "x": 921, "y": 5.5959332473777685 }, { "x": 922,  
"y": 1.695520870538198 }, { "x": 923, "y": 5.411006085106532 }, { "x": 924,  
"y": 2.3905154744037755 }, { "x": 925, "y": -6.198713151973552 }, { "x":  
926, "y": 8.846201028445464 }, { "x": 927, "y": -7.75792952524678 }, { "x":  
928, "y": 0.8385581116389176 }, { "x": 929, "y": 3.5706792893176136 }, {  
"x": 930, "y": 4.811035371914025 }, { "x": 931, "y": 8.25501647766464 }, {  
"x": 932, "y": -4.505711907612193 }, { "x": 933, "y": -7.732830378386682 },  
{ "x": 934, "y": -4.096858819010951 }, { "x": 935, "y": -1.7724247601789642  
}, { "x": 936, "y": 2.454224480590149 }, { "x": 937, "y": -8.375446987152106  
}, { "x": 938, "y": 2.7879096219650634 }, { "x": 939, "y":  
1.4043407763429343 }, { "x": 940, "y": -7.7866069543359355 }, { "x": 941,  
"y": -5.536339681781268 }, { "x": 942, "y": -0.9674248954225462 }, { "x":  
943, "y": 5.095492336205794 }, { "x": 944, "y": -5.486463805973028 }, { "x":  
945, "y": -7.9590647949470945 }, { "x": 946, "y": -8.321035881022578 }, {  
"x": 947, "y": 7.316036085318178 }, { "x": 948, "y": 3.835272089268237 }, {  
"x": 949, "y": -4.414426537965296 }, { "x": 950, "y": -2.4012622855936705 },  
{ "x": 951, "y": 3.3262475287984845 }, { "x": 952, "y": 6.857080381041797 },  
{ "x": 953, "y": 3.8400692297617116 }, { "x": 954, "y": 8.009268491019029 },  
{ "x": 955, "y": -0.5278174361013832 }, { "x": 956, "y": 8.018226706061476  
}, { "x": 957, "y": -5.9918969508950255 }, { "x": 958, "y":  
6.752605437338856 }, { "x": 959, "y": 0.42230908830922687 }, { "x": 960,  
"y": 6.250052860263313 }, { "x": 961, "y": 8.943807483524687 }, { "x": 962,  
"y": 4.523461953136595 }, { "x": 963, "y": -8.850548945862075 }, { "x": 964,  
"y": 0.4272550117331253 }, { "x": 965, "y": 3.7407657438971835 }, { "x":  
966, "y": 3.182227747001628 }, { "x": 967, "y": -3.9591757377893737 }, {  
"x": 968, "y": -0.47116203443763993 }, { "x": 969, "y": 0.1541985601228575  
}, { "x": 970, "y": 2.1468366586121768 }, { "x": 971, "y": 6.462550403389489  
}, { "x": 972, "y": 4.4178492663349935 }, { "x": 973, "y": -  
0.8684111130716605 }, { "x": 974, "y": 0.37282980442516767 }, { "x": 975,  
"y": -2.2906801798134424 }, { "x": 976, "y": 6.531636056325809 }, { "x":  
977, "y": -3.4515699813146528 }, { "x": 978, "y": -7.654248823761698 }, {  
"x": 979, "y": 2.9613260539018427 }, { "x": 980, "y": -5.770471040786519 },  
{ "x": 981, "y": -3.51023957795727 }, { "x": 982, "y": -0.45527149890988916  
}, { "x": 983, "y": -7.097038849540448 }, { "x": 984, "y": 6.090939840217295  
}, { "x": 985, "y": -0.36403645344384117 }, { "x": 986, "y":  
4.393531473842586 }, { "x": 987, "y": 0.4220729694886991 }, { "x": 988, "y":  
6.876724398922125 }, { "x": 989, "y": -2.0455276534682145 }, { "x": 990,  
"y": -1.5087449036808653 }, { "x": 991, "y": -7.760297797746748 }, { "x":  
992, "y": 1.2090194774185168 }, { "x": 993, "y": -0.6720715333543499 }, {
```

```

"x": 994, "y": 4.419107472905733 }, { "x": 995, "y": 7.895496845217107 }, {
"x": 996, "y": -4.458505036148511 }, { "x": 997, "y": 7.08009051513438 }, {
"x": 998, "y": 2.208353301296059 }, { "x": 999, "y": 1.6204274477434097 }, {
"x": 1000, "y": -7.001185487467807 }, { "x": 1001, "y": 6.690153112896867 },
{ "x": 1002, "y": -8.581816199680938 }, { "x": 1003, "y": 6.911939380124579
}, { "x": 1004, "y": -1.0476288088921324 }, { "x": 1005, "y": -
7.779252631295618 }, { "x": 1006, "y": -3.712719791381671 }, { "x": 1007,
"y": -5.317308933928048 }, { "x": 1008, "y": -8.974537261281622 }, { "x":
1009, "y": -3.0601572073217547 }, { "x": 1010, "y": -5.7292725297488545 }, {
"x": 1011, "y": -4.355087279810069 }, { "x": 1012, "y": -8.552984296037835
}, { "x": 1013, "y": 7.690199361546981 }, { "x": 1014, "y": -2.473613151824
}, { "x": 1015, "y": 3.1207289819764306 }, { "x": 1016, "y":
6.1202884665667145 }, { "x": 1017, "y": -4.287798170129186 }, { "x": 1018,
"y": -4.080728201420758 }, { "x": 1019, "y": -1.332595635264033 }, { "x":
1020, "y": 4.319295468019533 }, { "x": 1021, "y": -1.2993505772809417 }, {
"x": 1022, "y": -8.132445154081985 }, { "x": 1023, "y": -3.40543517438502 },
{ "x": 1024, "y": -8.02925211345205 }, { "x": 1025, "y": -2.758033796845541
}, { "x": 1026, "y": -2.5510237793648267 }, { "x": 1027, "y": -
0.6148063929366554 }, { "x": 1028, "y": 2.830555597761361 }, { "x": 1029,
"y": -4.7964615718854375 },
{ "x": 1030, "y": -2.2641818090965975 }, { "x": 1031, "y":
0.40261219409392623 }, { "x": 1032, "y": -6.641958508748897 }, { "x": 1033,
"y": -2.2342934865958615 }, { "x": 1034, "y": 1.3280643056323225 }, { "x":
1035, "y": 2.275348639787236 }, { "x": 1036, "y": -3.8968881856865423 }, {
"x": 1037, "y": -8.656741785237703 }, { "x": 1038, "y": 4.612952229745904 },
{ "x": 1039, "y": -4.572700776172702 }, { "x": 1040, "y": 8.98932695981949
}, { "x": 1041, "y": 0.617706918271665 }, { "x": 1042, "y":
4.357539924049943 }, { "x": 1043, "y": -1.393741575958174 }, { "x": 1044,
"y": 1.2491374619481483 }, { "x": 1045, "y": -8.729047551893162 }, { "x":
1046, "y": 1.4871304642642578 }, { "x": 1047, "y": 8.155789506716097 }, {
"x": 1048, "y": -6.77001899024159 }, { "x": 1049, "y": 6.2830042519145195 },
{ "x": 1050, "y": -4.63952428085407 }, { "x": 1051, "y": 4.205775321487813
}, { "x": 1052, "y": 3.576380714160747 }, { "x": 1053, "y": -
1.5910362276157102 }, { "x": 1054, "y": -7.279912736633798 }, { "x": 1055,
"y": -3.027278105985994 }, { "x": 1056, "y": -7.340578746023818 }, { "x":
1057, "y": -4.205525574994663 }, { "x": 1058, "y": -7.418839346158628 }, {
"x": 1059, "y": -3.4732758965754957 }, { "x": 1060, "y": -2.357376281528593
}, { "x": 1061, "y": -5.92343495394344 }, { "x": 1062, "y": -
8.060253798886675 }, { "x": 1063, "y": 0.31537335543121614 }, { "x": 1064,
"y": -8.219036202821428 }, { "x": 1065, "y": -2.271882929308947 }, { "x":
1066, "y": 4.64563662978685 }, { "x": 1067, "y": -4.697460735985084 }, {
"x": 1068, "y": -5.121657125265537 }, { "x": 1069, "y": 1.9350540922155304
}, { "x": 1070, "y": 8.405730534520472 }, { "x": 1071, "y": -
5.461573286753498 }, { "x": 1072, "y": -0.6077999410727024 }, { "x": 1073,
"y": 3.6152764945076665 }, { "x": 1074, "y": 3.1601001459701425 }, { "x":
1075, "y": -4.156850286487872 }, { "x": 1076, "y": 8.913394880905923 }, {
"x": 1077, "y": -2.5433159009567143 }, { "x": 1078, "y": -8.198039499865141
}, { "x": 1079, "y": -7.945069226163152 }, { "x": 1080, "y":
1.6431847037987168 }, { "x": 1081, "y": -7.003881776548704 }, { "x": 1082,
"y": 4.57166495549251 }, { "x": 1083, "y": 4.263994321112657 }, { "x": 1084,
"y": 7.242362603221981 }, { "x": 1085, "y": 7.107520748154236 }, { "x":
1086, "y": 3.029537187061001 }, { "x": 1087, "y": 3.6544732877930066 }, {
"x": 1088, "y": -5.665528785778358 }, { "x": 1089, "y": 6.097541918870785 },
{ "x": 1090, "y": 8.851128861723467 }, { "x": 1091, "y": 3.6896759496349762
}, { "x": 1092, "y": 0.6740965251781965 }, { "x": 1093, "y":
1.0362761746354927 }, { "x": 1094, "y": -0.2755433719633711 }, { "x": 1095,
"y": 6.393646339409241 }, { "x": 1096, "y": 0.778941777879016 }, { "x":

```

```

1097, "y": 1.7249388890704331 }, { "x": 1098, "y": 2.6582101368332687 }, {
"x": 1099, "y": -0.7812946822473084 }, { "x": 1100, "y": -
0.09469338147651207 }, { "x": 1101, "y": 1.8292269876807303 }, { "x": 1102,
"y": -4.928713851233835 }, { "x": 1103, "y": -8.886571237361961 }, { "x":
1104, "y": -2.6912526042763716 }, { "x": 1105, "y": 6.574998783773577 }, {
"x": 1106, "y": -8.242703908160298 }, { "x": 1107, "y": -0.0815355870079788
}, { "x": 1108, "y": -1.143564945899504 }, { "x": 1109, "y": -
0.8082302611052086 }, { "x": 1110, "y": 6.2631957117851975 }, { "x": 1111,
"y": -8.948779085243926 }, { "x": 1112, "y": -2.4342838124844803 }, { "x":
1113, "y": -1.4367537715845806 }, { "x": 1114, "y": -5.442948781300114 }, {
"x": 1115, "y": -0.679379543448217 }, { "x": 1116, "y": -4.572905708197316
}, { "x": 1117, "y": -3.662564965340544 }, { "x": 1118, "y": -
1.0851380359264393 }, { "x": 1119, "y": -6.176112575321746 }, { "x": 1120,
"y": -2.678578388281263 }, { "x": 1121, "y": 6.51879932847805 }, { "x":
1122, "y": 8.640243885405607 }, { "x": 1123, "y": -3.33200212097678 }, {
"x": 1124, "y": 0.048312307661953824 }, { "x": 1125, "y": -
2.3339691097810036 }, { "x": 1126, "y": -8.89122060000997 }, { "x": 1127,
"y": -3.5650268471781583 }, { "x": 1128, "y": 8.088945327453288 }, { "x":
1129, "y": 8.020496531450725 }, { "x": 1130, "y": 0.37048498481813574 }, {
"x": 1131, "y": -7.522235046282367 }, { "x": 1132, "y": -7.929361675568241
}, { "x": 1133, "y": 0.70473124282179 }, { "x": 1134, "y": -
3.8448397667180068 }, { "x": 1135, "y": 7.725778665082704 }, { "x": 1136,
"y": -2.5712127674828604 }, { "x": 1137, "y": 5.208599896671192 }, { "x":
1138, "y": 4.510639395770131 }, { "x": 1139, "y": -5.369979375723929 }, {
"x": 1140, "y": -2.551335117119656 }, { "x": 1141, "y": -0.6970286238824741
}, { "x": 1142, "y": -3.672224378625847 }, { "x": 1143, "y": -
7.044444003155402 }, { "x": 1144, "y": 0.04455431356552886 }, { "x": 1145,
"y": -6.841007302855261 }, { "x": 1146, "y": 5.666600646971162 }, { "x":
1147, "y": 3.5287989046710564 }, { "x": 1148, "y": 0.48879266403550936 }, {
"x": 1149, "y": -6.862757563114394 }, { "x": 1150, "y": 5.131070811216844 },
{ "x": 1151, "y": 0.889595072765033 }, { "x": 1152, "y": 4.7299115061109305
}, { "x": 1153, "y": 7.398488088859825 }, { "x": 1154, "y": -
7.486327466853863 }, { "x": 1155, "y": -0.642993005423385 }, { "x": 1156,
"y": 6.9703606408873195 }, { "x": 1157, "y": -5.437883405134295 }, { "x":
1158, "y": 3.7891939952195983 }, { "x": 1159, "y": -2.9772159065486665 }, {
"x": 1160, "y": 7.459513035803148 }, { "x": 1161, "y": -3.382933032471226 },
{ "x": 1162, "y": -5.335627048077868 }, { "x": 1163, "y": -5.442804436456887
}, { "x": 1164, "y": -8.635897605787324 }, { "x": 1165, "y": -
0.6636487740326764 }, { "x": 1166, "y": 5.569983093748242 }, { "x": 1167,
"y": 3.7439838789817266 }, { "x": 1168, "y": 6.5536873053527955 }, { "x":
1169, "y": 0.7352640540141255 }, { "x": 1170, "y": 4.624573777147264 }, {
"x": 1171, "y": 4.360033664187371 }, { "x": 1172, "y": 0.22516820261424897
}, { "x": 1173, "y": 5.075531153084501 }, { "x": 1174, "y": -
2.9281330039027242 }, { "x": 1175, "y": 0.566430857072298 }, { "x": 1176,
"y": -6.012068410851321 }, { "x": 1177, "y": -3.9889578681175664 }, { "x":
1178, "y": -2.4664860364244037 }, { "x": 1179, "y": 4.923217962524266 }, {
"x": 1180, "y": -4.010289016272933 }, { "x": 1181, "y": -8.275860338742074
}, { "x": 1182, "y": -0.6370177151315701 }, { "x": 1183, "y": -
2.4649476681633136 }, { "x": 1184, "y": -2.817200320393786 }, { "x": 1185,
"y": 4.959331089946609 }, { "x": 1186, "y": -1.2793643753851018 }, { "x":
1187, "y": 8.42662602806545 }, { "x": 1188, "y": -0.5431068046690761 }, {
"x": 1189, "y": 1.6861440796157208 }, { "x": 1190, "y": 1.2129582082119796
}, { "x": 1191, "y": 1.8702520106042009 }, { "x": 1192, "y": -
6.054972838512857 }, { "x": 1193, "y": 1.9640160973757776 }, { "x": 1194,
"y": -6.162940842090109 }, { "x": 1195, "y": -1.8871124874962693 }, { "x":
1196, "y": -0.3420541501465255 }, { "x": 1197, "y": 1.987749715112935 }, {
"x": 1198, "y": 7.498636837896299 }, { "x": 1199, "y": -4.695845683538796 },

```



```

{ "x": 1200, "y": -7.349229626165894 }, { "x": 1201, "y": 5.774516995839516
}, { "x": 1202, "y": 6.6724393309524945 }, { "x": 1203, "y":
5.909283786332695 }, { "x": 1204, "y": -5.006083068191179 }, { "x": 1205,
"y": 7.543592754792218 }, { "x": 1206, "y": 4.952677672344912 }, { "x":
1207, "y": -5.285309812657111 }, { "x": 1208, "y": -2.6457810543057496 }, {
"x": 1209, "y": -6.952198967144237 }, { "x": 1210, "y": -6.9190642371659745
}, { "x": 1211, "y": 4.253023777938674 }, { "x": 1212, "y": -
0.07783583643259995 }, { "x": 1213, "y": 6.324449530570215 }, { "x": 1214,
"y": -1.6749133382993069 }, { "x": 1215, "y": 0.44922112452580265 }, { "x":
1216, "y": -5.749150331931404 }, { "x": 1217, "y": -2.0474108645708986 }, {
"x": 1218, "y": 7.822820180216993 }, { "x": 1219, "y": 7.407058370156136 },
{ "x": 1220, "y": -1.8873430520640593 }, { "x": 1221, "y":
7.2301291821364195 }, { "x": 1222, "y": 7.075284155318169 }, { "x": 1223,
"y": -3.54396133188272 }, { "x": 1224, "y": -8.109885020119549 }, { "x":
1225, "y": 6.9264542472248 }, { "x": 1226, "y": 3.681201739181006 }, { "x":
1227, "y": -7.8526362228683695 }, { "x": 1228, "y": 2.748704221096858 }, {
"x": 1229, "y": 2.3922838561875377 }, { "x": 1230, "y": 8.395146733238175 },
{ "x": 1231, "y": -6.52299829465362 }, { "x": 1232, "y": -8.917474149664507
}, { "x": 1233, "y": -3.8110940953078316 }, { "x": 1234, "y":
0.5006027966341904 }, { "x": 1235, "y": 7.056930898730354 }, { "x": 1236,
"y": -1.543851536201796 }, { "x": 1237, "y": 5.843926973329673 }, { "x":
1238, "y": 7.022028253447523 }, { "x": 1239, "y": 3.3639592153273643 }, {
"x": 1240, "y": -4.973479718222883 }, { "x": 1241, "y": -1.7337006331402032
}, { "x": 1242, "y": 8.192038535254845 }, { "x": 1243, "y":
4.909576183412867 }, { "x": 1244, "y": -2.8991366287012443 }, { "x": 1245,
"y": 8.7880377613794 }, { "x": 1246, "y": -3.8780045566344876 }, { "x":
1247, "y": -4.277475432012289 }, { "x": 1248, "y": -2.4841604063697016 }, {
"x": 1249, "y": -2.876923906423343 }, { "x": 1250, "y": 7.229677521449354 },
{ "x": 1251, "y": -1.309184370707536 }, { "x": 1252, "y": -
7.9083398227660116 }, { "x": 1253, "y": -5.180782561731823 }, { "x": 1254,
"y": -6.778568424046892 }, { "x": 1255, "y": 8.326806035811721 }, { "x":
1256, "y": 5.618217549280349 }, { "x": 1257, "y": 1.624454531006693 }, {
"x": 1258, "y": -1.0510715307240464 }, { "x": 1259, "y": 5.455995447975546
}, { "x": 1260, "y": -4.347032557696544 }, { "x": 1261, "y":
4.190999230044358 }, { "x": 1262, "y": 8.18313872495354 }, { "x": 1263, "y":
-6.739126459922138 }, { "x": 1264, "y": -6.213744980189357 }, { "x": 1265,
"y": -6.804420740634686 }, { "x": 1266, "y": -1.9356550537621597 }, { "x":
1267, "y": 6.054749273021452 }, { "x": 1268, "y": -4.122520997125104 }, {
"x": 1269, "y": 0.5496770676225307 }, { "x": 1270, "y": 2.3501373557514764
}, { "x": 1271, "y": -6.0966008755488375 }, { "x": 1272, "y": -
0.7850868957419124 }, { "x": 1273, "y": -3.045442265727301 }, { "x": 1274,
"y": 1.9987451106495726 }, { "x": 1275, "y": -4.05852007614016 }, { "x":
1276, "y": -8.193451557394457 }, { "x": 1277, "y": 3.2270855677224812 }, {
"x": 1278, "y": -4.616644770714548 }, { "x": 1279, "y": 5.095262343002446 },
{ "x": 1280, "y": -0.29254198291304334 }, { "x": 1281, "y": -
0.11463455554362234 }, { "x": 1282, "y": -2.809969534180852 }, { "x": 1283,
"y": 7.057831246352556 }, { "x": 1284, "y": -6.1479308039136 }, { "x": 1285,
"y": -5.792273539232811 }, { "x": 1286, "y": 6.465071878279568 }, { "x":
1287, "y": -1.4653216531275142 }, { "x": 1288, "y": 4.597103356939506 }, {
"x": 1289, "y": 7.758099173289853 }, { "x": 1290, "y": 0.9213294271386978 },
{ "x": 1291, "y": -3.3363049460787213 }, { "x": 1292, "y": 4.914068903893565
}, { "x": 1293, "y": -0.10888284071522492 }, { "x": 1294, "y":
8.420031948779808 }, { "x": 1295, "y": 3.675826353590825 }, { "x": 1296,
"y": 2.6369610040234015 }, { "x": 1297, "y": 1.2560731181266966 }, { "x":
1298, "y": -3.601006212426431 }, { "x": 1299, "y": -2.1098583599212963 }, {
"x": 1300, "y": -8.103270722817463 }, { "x": 1301, "y": 1.7751158662161188
}, { "x": 1302, "y": -8.969462333139957 }, { "x": 1303, "y": -

```

```
5.970230992435614 }, { "x": 1304, "y": 0.3416827141980665 }, { "x": 1305,
"y": -7.1257974199777 }, { "x": 1306, "y": 7.441004563977984 }, { "x": 1307,
"y": -6.987693206165195 }, { "x": 1308, "y": -2.6228792220575627 }, { "x":
1309, "y": -6.439813505443322 }, { "x": 1310, "y": 1.5223787496072756 }, {
"x": 1311, "y": 0.9574311067245969 }, { "x": 1312, "y": 6.8940925113476705
}, { "x": 1313, "y": 5.978780641585292 }, { "x": 1314, "y": 7.27624471902773
}, { "x": 1315, "y": -3.358440290388109 }, { "x": 1316, "y":
4.6468100844530795 }, { "x": 1317, "y": 3.527298241854645 }, { "x": 1318,
"y": 0.5277985834198802 }, { "x": 1319, "y": 1.703571105491136 }, { "x":
1320, "y": 5.498877851347489 }, { "x": 1321, "y": 0.8743998235496271 }, {
"x": 1322, "y": 5.672339002311247 }, { "x": 1323, "y": -5.592441362908504 },
{ "x": 1324, "y": -6.04141661423844 }, { "x": 1325, "y": -3.8201773294641823
}, { "x": 1326, "y": -0.37015107165923666 }, { "x": 1327, "y":
1.105376394637947 }, { "x": 1328, "y": 5.716517275745426 }, { "x": 1329,
"y": -6.569394022676612 }, { "x": 1330, "y": -7.323917861677879 }, { "x":
1331, "y": 8.244339645150127 }, { "x": 1332, "y": -2.0654779669524714 }, {
"x": 1333, "y": 7.68368227719764 }, { "x": 1334, "y": 7.757350547145563 }, {
"x": 1335, "y": 7.281411894983542 }, { "x": 1336, "y": 6.9581499737044545 },
{ "x": 1337, "y": 4.3326596022350135 }, { "x": 1338, "y": -5.77295612637632
}, { "x": 1339, "y": 8.500758094295989 }, { "x": 1340, "y":
4.272980861330227 }, { "x": 1341, "y": -2.0754436616666974 }, { "x": 1342,
"y": 2.2007294460287046 }, { "x": 1343, "y": 8.933642167603494 }, { "x":
1344, "y": 8.7628039158596 }, { "x": 1345, "y": 0.4754106422542872 }, { "x":
1346, "y": 2.4441753528211905 }, { "x": 1347, "y": 0.26562592946002894 }, {
"x": 1348, "y": 6.516059855901849 }, { "x": 1349, "y": -6.188237133169256 },
{ "x": 1350, "y": -1.2772924204716176 }, { "x": 1351, "y": -
5.286072858836642 }, { "x": 1352, "y": -5.495319679612326 }, { "x": 1353,
"y": -1.1213512149073992 }, { "x": 1354, "y": 7.524320842269933 }, { "x":
1355, "y": -3.6668247361022477 }, { "x": 1356, "y": -0.17375379029086346 },
{ "x": 1357, "y": 2.241922609162973 }, { "x": 1358, "y": -3.1922134823535746
}, { "x": 1359, "y": 1.6734095071072286 }, { "x": 1360, "y": -
1.9960556355367123 }, { "x": 1361, "y": -2.3560917418322536 }, { "x": 1362,
"y": -1.4630069608547727 }, { "x": 1363, "y": 8.151526685527372 }, { "x":
1364, "y": 5.208621003820216 }, { "x": 1365, "y": 0.6273937763591348 }, {
"x": 1366, "y": 5.053718490393038 }, { "x": 1367, "y": 2.385378455643149 },
{ "x": 1368, "y": -5.367075917406086 }, { "x": 1369, "y": -8.218106891824581
}, { "x": 1370, "y": -1.3691425418590963 }, { "x": 1371, "y": -
3.792211866173907 }, { "x": 1372, "y": 2.863930940228901 }, { "x": 1373,
"y": 3.313000853530008 }, { "x": 1374, "y": 3.739314145316767 }, { "x":
1375, "y": -7.791437295262062 }, { "x": 1376, "y": -7.788438232330342 }, {
"x": 1377, "y": 8.8638038476467 }, { "x": 1378, "y": -3.6529741919146055 },
{ "x": 1379, "y": -2.3939596571291837 }, { "x": 1380, "y": 5.310316486700142
}, { "x": 1381, "y": 4.987877328598099 }, { "x": 1382, "y":
4.742458401405173 }, { "x": 1383, "y": 2.9193949813696207 }, { "x": 1384,
"y": -3.4582340071180706 }, { "x": 1385, "y": -0.6926030841868034 }, { "x":
1386, "y": 7.422611799645068 }, { "x": 1387, "y": 6.758886691461759 }, {
"x": 1388, "y": -2.924044759346984 }, { "x": 1389, "y": -0.7112558176461832
}, { "x": 1390, "y": 8.678647428844496 }, { "x": 1391, "y":
5.921663806271024 }, { "x": 1392, "y": 3.860176421061551 }, { "x": 1393,
"y": -6.07711681183257 }, { "x": 1394, "y": 3.3485173510070183 }, { "x":
1395, "y": -3.926647127172341 }, { "x": 1396, "y": 2.9180874228200064 }, {
"x": 1397, "y": 3.4456779564391713 }, { "x": 1398, "y": 4.306034762080882 },
{ "x": 1399, "y": -7.681191899904279 }, { "x": 1400, "y": -5.517077309911213
}, { "x": 1401, "y": 1.4488235622604257 }, { "x": 1402, "y": -
8.87010460053737 }, { "x": 1403, "y": 7.378785948540187 }, { "x": 1404, "y":
-1.6303700043056768 }, { "x": 1405, "y": -6.9129868652249655 }, { "x": 1406,
"y": -7.321818240312066 }, { "x": 1407, "y": -0.24168320196285187 }, { "x":
1408, "y": 1.105376394637947 }, { "x": 1409, "y": 5.716517275745426 }, { "x":
1410, "y": -6.569394022676612 }, { "x": 1411, "y": -7.323917861677879 }, { "x":
1412, "y": 8.244339645150127 }, { "x": 1413, "y": -2.0654779669524714 }, {
"x": 1414, "y": 7.68368227719764 }, { "x": 1415, "y": 7.757350547145563 }, {
"x": 1416, "y": 7.281411894983542 }, { "x": 1417, "y": 6.9581499737044545 },
{ "x": 1418, "y": 4.3326596022350135 }, { "x": 1419, "y": -5.77295612637632
}, { "x": 1420, "y": 8.500758094295989 }, { "x": 1421, "y":
4.272980861330227 }, { "x": 1422, "y": -2.0754436616666974 }, { "x": 1423,
"y": 2.2007294460287046 }, { "x": 1424, "y": 8.933642167603494 }, { "x":
1425, "y": 8.7628039158596 }, { "x": 1426, "y": 0.4754106422542872 }, { "x":
1427, "y": 2.4441753528211905 }, { "x": 1428, "y": 0.26562592946002894 }, {
"x": 1429, "y": 6.516059855901849 }, { "x": 1430, "y": -6.188237133169256 },
{ "x": 1431, "y": -1.2772924204716176 }, { "x": 1432, "y": -
5.286072858836642 }, { "x": 1433, "y": -5.495319679612326 }, { "x": 1434,
"y": -1.1213512149073992 }, { "x": 1435, "y": 7.524320842269933 }, { "x":
1436, "y": -3.6668247361022477 }, { "x": 1437, "y": -0.17375379029086346 },
{ "x": 1438, "y": 2.241922609162973 }, { "x": 1439, "y": -3.1922134823535746
}, { "x": 1440, "y": 1.6734095071072286 }, { "x": 1441, "y": -
1.9960556355367123 }, { "x": 1442, "y": -2.3560917418322536 }, { "x": 1443,
"y": -1.4630069608547727 }, { "x": 1444, "y": 8.151526685527372 }, { "x":
1445, "y": 5.208621003820216 }, { "x": 1446, "y": 0.6273937763591348 }, {
"x": 1447, "y": 5.053718490393038 }, { "x": 1448, "y": 2.385378455643149 },
{ "x": 1449, "y": -5.367075917406086 }, { "x": 1450, "y": -8.218106891824581
}, { "x": 1451, "y": -1.3691425418590963 }, { "x": 1452, "y": -
3.792211866173907 }, { "x": 1453, "y": 2.863930940228901 }, { "x": 1454,
"y": 3.313000853530008 }, { "x": 1455, "y": 3.739314145316767 }, { "x":
1456, "y": -7.791437295262062 }, { "x": 1457, "y": -7.788438232330342 }, {
"x": 1458, "y": 8.8638038476467 }, { "x": 1459, "y": -3.6529741919146055 },
{ "x": 1460, "y": -2.3939596571291837 }, { "x": 1461, "y": 5.310316486700142
}, { "x": 1462, "y": 4.987877328598099 }, { "x": 1463, "y":
4.742458401405173 }, { "x": 1464, "y": 2.9193949813696207 }, { "x": 1465,
"y": -3.4582340071180706 }, { "x": 1466, "y": -0.6926030841868034 }, { "x":
1467, "y": 7.422611799645068 }, { "x": 1468, "y": 6.758886691461759 }, {
"x": 1469, "y": -2.924044759346984 }, { "x": 1470, "y": -0.7112558176461832
}, { "x": 1471, "y": 8.678647428844496 }, { "x": 1472, "y":
5.921663806271024 }, { "x": 1473, "y": 3.860176421061551 }, { "x": 1474,
"y": -6.07711681183257 }, { "x": 1475, "y": 3.3485173510070183 }, { "x":
1476, "y": -3.926647127172341 }, { "x": 1477, "y": 2.9180874228200064 }, {
"x": 1478, "y": 3.4456779564391713 }, { "x": 1479, "y": 4.306034762080882 },
{ "x": 1480, "y": -7.681191899904279 }, { "x": 1481, "y": -5.517077309911213
}, { "x": 1482, "y": 1.4488235622604257 }, { "x": 1483, "y": -
8.87010460053737 }, { "x": 1484, "y": 7.378785948540187 }, { "x": 1485, "y":
-1.6303700043056768 }, { "x": 1486, "y": -6.9129868652249655 }, { "x": 1487,
"y": -7.321818240312066 }, { "x": 1488, "y": -0.24168320196285187 }, { "x":
1489, "y": 1.105376394637947 }, { "x": 1490, "y": 5.716517275745426 }, { "x":
1491, "y": -6.569394022676612 }, { "x": 1492, "y": -7.323917861677879 }, { "x":
1493, "y": 8.244339645150127 }, { "x": 1494, "y": -2.0654779669524714 }, {
"x": 1495, "y": 7.68368227719764 }, { "x": 1496, "y": 7.757350547145563 }, {
"x": 1497, "y": 7.281411894983542 }, { "x": 1498, "y": 6.9581499737044545 },
{ "x": 1499, "y": 4.3326596022350135 }, { "x": 1500, "y": -5.77295612637632
}, { "x": 1501, "y": 8.500758094295989 }, { "x": 1502, "y":
4.272980861330227 }, { "x": 1503, "y": -2.0754436616666974 }, { "x": 1504,
"y": 2.2007294460287046 }, { "x": 1505, "y": 8.933642167603494 }, { "x":
1506, "y": 8.7628039158596 }, { "x": 1507, "y": 0.4754106422542872 }, { "x":
1508, "y": 2.4441753528211905 }, { "x": 1509, "y": 0.26562592946002894 }, {
"x": 1510, "y": 6.516059855901849 }, { "x": 1511, "y": -6.188237133169256 },
{ "x": 1512, "y": -1.2772924204716176 }, { "x": 1513, "y": -
5.286072858836642 }, { "x": 1514, "y": -5.495319679612326 }, { "x": 1515,
"y": -1.1213512149073992 }, { "x": 1516, "y": 7.524320842269933 }, { "x":
1517, "y": -3.6668247361022477 }, { "x": 1518, "y": -0.17375379029086346 },
{ "x": 1519, "y": 2.241922609162973 }, { "x": 1520, "y": -3.1922134823535746
}, { "x": 1521, "y": 1.6734095071072286 }, { "x": 1522, "y": -
1.9960556355367123 }, { "x": 1523, "y": -2.3560917418322536 }, { "x": 1524,
"y": -1.4630069608547727 }, { "x": 1525, "y": 8.151526685527372 }, { "x":
1526, "y": 5.208621003820216 }, { "x": 1527, "y": 0.6273937763591348 }, {
"x": 1528, "y": 5.053718490393038 }, { "x": 1529, "y": 2.385378455643149 },
{ "x": 1530, "y": -5.367075917406086 }, { "x": 1531, "y": -8.218106891824581
}, { "x": 1532, "y": -1.3691425418590963 }, { "x": 1533, "y": -
3.792211866173907 }, { "x": 1534, "y": 2.863930940228901 }, { "x": 1535,
"y": 3.313000853530008 }, { "x": 1536, "y": 3.739314145316767 }, { "x":
1537, "y": -7.791437295262062 }, { "x": 1538, "y": -7.788438232330342 }, {
"x": 1539, "y": 8.8638038476467 }, { "x": 1540, "y": -3.6529741919146055 },
{ "x": 1541, "y": -2.3939596571291837 }, { "x": 1542, "y": 5.310316486700142
}, { "x": 1543, "y": 4.987877328598099 }, { "x": 1544, "y":
4.742458401405173 }, { "x": 1545, "y": 2.9193949813696207 }, { "x": 1546,
"y": -3.4582340071180706 }, { "x": 1547, "y": -0.6926030841868034 }, { "x":
1548, "y": 7.422611799645068 }, { "x": 1549, "y": 6.758886691461759 }, {
"x": 1550, "y": -2.924044759346984 }, { "x": 1551, "y": -0.7112558176461832
}, { "x": 1552, "y": 8.678647428844496 }, { "x": 1553, "y":
5.921663806271024 }, { "x": 1554, "y": 3.860176421061551 }, { "x": 1555,
"y": -6.07711681183257 }, { "x": 1556, "y": 3.3485173510070183 }, { "x":
1557, "y": -3.926647127172341 }, { "x": 1558, "y": 2.9180874228200064 }, {
"x": 1559, "y": 3.4456779564391713 }, { "x": 1560, "y": 4.306034762080882 },
{ "x": 1561, "y": -7.681191899904279 }, { "x": 1562, "y": -5.517077309911213
}, { "x": 1563, "y": 1.4488235622604257 }, { "x": 1564, "y": -
8.87010460053737 }, { "x": 1565, "y": 7.378785948540187 }, { "x": 1566, "y":
-1.6303700043056768 }, { "x": 1567, "y": -6.9129868652249655 }, { "x": 1568,
"y": -7.321818240312066 }, { "x": 1569, "y": -0.24168320196285187 }, { "x":
1570, "y": 1.105376394637947 }, { "x": 1571, "y": 5.716517275745426 }, { "x":
1572, "y": -6.569394022676612 }, { "x": 1573, "y": -7.323917861677879 }, { "x":
1574, "y": 8.244339645150127 }, { "x": 1575, "y": -2.0654779669524714 }, {
"x": 1576, "y": 7.68368227719764 }, { "x": 1577, "y": 7.757350547145563 }, {
"x": 1578, "y": 7.281411894983542 }, { "x": 1579, "y": 6.9581499737044545 },
{ "x": 1580, "y": 4.3326596022350135 }, { "x": 1581, "y": -5.77295612637632
}, { "x": 1582, "y": 8.500758094295989 }, { "x": 1583, "y":
4.272980861330227 }, { "x": 1584, "y": -2.0754436616666974 }, { "x": 1585,
"y": 2.2007294460287046 }, { "x": 1586, "y": 8.933642167603494 }, { "x":
1587, "y": 8.7628039158596 }, { "x": 1588, "y": 0.4754106422542872 }, { "x":
1589, "y": 2.4441753528211905 }, { "x": 1590, "y": 0.26562592946002894 }, {
"x": 1591, "y": 6.516059855901849 }, { "x": 1592, "y": -6.188237133169256 },
{ "x": 1593, "y": -1.2772924204716176 }, { "x": 1594, "y": -
5.286072858836642 }, { "x": 1595, "y": -5.495319679612326 }, { "x": 1596,
"y": -1.1213512149073992 }, { "x": 1597, "y": 7.524320842269933 }, { "x":
1598, "y": -3.6668247361022477 }, { "x": 1599, "y": -0.17375379029086346 },
{ "x": 1600, "y": 2.241922609162973 }, { "x": 1601, "y": -3.1922134823535746
}, { "x": 1602, "y": 1.6734095071072286 }, { "x": 1603, "y": -
1.9960556355367123 }, { "x": 1604, "y": -2.3560917418322536 }, { "x": 1605,
"y": -1.4630069608547727 }, { "x": 1606, "y": 8.151526685527372 }, { "x":
1607, "y": 5.208621003820216 }, { "x": 1608, "y": 0.6273937763591348 }, {
"x": 1609, "y": 5.053718490393038 }, { "x": 1610, "y": 2.385378455643149 },
{ "x": 1611, "y": -5.367075917406086 }, { "x": 1612, "y": -8.218106891824581
}, { "x": 1613, "y": -1.3691425418590963 }, { "x": 1614, "y": -
3.792211866173907 }, { "x": 1615, "y": 2.863930940228901 }, { "x": 1616,
"y": 3.313000853530008 }, { "x": 1617, "y": 3.739314145316767 }, { "x":
1618, "y": -7.791437295262062 }, { "x": 1619, "y": -7.788438232330342 }, {
"x": 1620, "y": 8.8638038476467 }, { "x": 1621, "y": -3.6529741919146055 },
{ "x": 1622, "y": -2.3939596571291837 }, { "x": 1623, "y": 5.310316486700142
}, { "x": 1624, "y": 4.987877328598099 }, { "x": 1625, "y":
4.742458401405173 }, { "x": 1626, "y": 2.9193949813696207 }, { "x": 1627,
"y": -3.4582340071180706 }, { "x": 1628, "y": -0.6926030841868034 }, { "x":
1629, "y": 7.422611799645068 }, { "x": 1630, "y": 6.758886691461759 }, {
"x": 1631, "y": -2.924044759346984 }, { "x": 1632, "y": -0.7112558176461832
}, { "x": 1633, "y": 8.678647428844496 }, { "x": 1634, "y":
5.921663806271024 }, { "x": 1635, "y": 3.860176421061551 }, { "x": 1636,
"y": -6.07711681183257 }, { "x": 1637, "y": 3.3485173510070183 }, { "x":
1638, "y": -3.926647127172341 }, { "x": 1639, "y": 2.9180874228200064 }, {
"x": 1640, "y": 3.4456779564391713 }, { "x": 1641, "y": 4.306034762080882 },
{ "x": 1642, "y": -7.681191899904279 }, { "x": 1643, "y": -5.517077309911213
}, { "x": 1644, "y": 1.4488235622604257 }, { "x": 1645, "y": -
8.87010460053737 }, { "x": 1646, "y": 7.378785948540187 }, { "x": 1647, "y":
-1.6303700043056768 }, { "x": 1648, "y": -6.9129868652249655 }, { "x": 1649,
"y": -7.321818240312066 }, { "x": 1650, "y": -0.24168320196285187 }, { "x":
1651, "y": 1.105376394637947 }, { "x": 1652, "y": 5.716517275745426 }, { "x":
1653, "y": -6.569394022676612 }, { "x": 1654, "y": -7.323917861677879 }, { "x":
1655, "y": 8.244339645150127 }, { "x": 1656, "y": -2.0654779669524714 }, {
"x": 1657, "y": 7.68368227719764 }, { "x": 1658, "y": 7.757350547145563 }, {
"x": 1659, "y": 7.281411894983542 }, { "x": 1660, "y": 6.9581499737044545 },
{ "x": 1661, "y": 4.3326596022350135 }, { "x": 1662, "y": -5.77295612637632
}, { "x": 1663, "y": 8.500758094295989 }, { "x": 1664, "y":
4.272980861330227 }, { "x": 1665, "y": -2.0754436616666974 }, { "x": 1666,
"y": 2.2007294460287046 }, { "x": 1667, "y": 8.933642167603494 }, { "x":
1668, "y": 8.7628039158596 }, { "x": 1669, "y": 0.4754106422542872 }, { "x":
1670, "y": 2.4441753528211905 }, { "x": 1671, "y": 0.26562592946002894 }, {
"x": 1672, "y": 6.516059855901849 }, { "x": 1673, "y": -6.188237133169256 },
{ "x": 1674, "y": -1.2772924204716176 }, { "x": 1675, "y": -
5.286072858836642 }, { "x": 1676, "y": -5.495319679612326 }, { "x": 1677,
"y": -1.1213512149073992 }, { "x": 1678, "y": 7.524320842269933 }, { "x":
1679, "y": -3.6668247361022477 }, { "x": 1680, "y":
```

```

1408, "y": -1.2646679433819035 }, { "x": 1409, "y": -8.925524767947476 }, {
"x": 1410, "y": 4.399454449493058 }, { "x": 1411, "y": 2.1169455476816808 },
{ "x": 1412, "y": 4.192761768133952 }, { "x": 1413, "y": 1.0061838735188537
}, { "x": 1414, "y": -4.647263936378101 }, { "x": 1415, "y": -
7.287923086907085 }, { "x": 1416, "y": -4.532156247795244 }, { "x": 1417,
"y": -6.482903445954879 }, { "x": 1418, "y": 8.558965226525835 }, { "x":
1419, "y": -7.1107576434220725 }, { "x": 1420, "y": 4.982209998861709 }, {
"x": 1421, "y": 4.0812200517324975 }, { "x": 1422, "y": -6.632814577495278
}, { "x": 1423, "y": 2.9783617870866976 }, { "x": 1424, "y":
1.1982781764595103 }, { "x": 1425, "y": -5.077370061188308 }, { "x": 1426,
"y": 3.035732862931612 }, { "x": 1427, "y": 7.705794331646253 }, { "x":
1428, "y": 0.87064889231522 }, { "x": 1429, "y": 8.852207299969635 }, { "x":
1430, "y": -0.38083925507052285 }, { "x": 1431, "y": -5.4573003373611915 },
{ "x": 1432, "y": -3.9503315609351892 }, { "x": 1433, "y": 6.481229058070399
}, { "x": 1434, "y": -2.8048100740261788 }, { "x": 1435, "y":
6.000939098065462 }, { "x": 1436, "y": 6.525546995905806 }, { "x": 1437,
"y": 6.186961130851177 }, { "x": 1438, "y": 3.0060989125082855 }, { "x":
1439, "y": 3.0029064602370976 }, { "x": 1440, "y": -7.053645138889621 }, {
"x": 1441, "y": 3.69479875482517 }, { "x": 1442, "y": -5.148522517605952 },
{ "x": 1443, "y": 0.7782333681557247 }, { "x": 1444, "y": -
0.48054077018756836 }, { "x": 1445, "y": 6.834423224704757 }, { "x": 1446,
"y": -2.798823414907007 }, { "x": 1447, "y": 7.715435022989968 }, { "x":
1448, "y": 5.9887019152372645 }, { "x": 1449, "y": 0.28157988309548365 }, {
"x": 1450, "y": -6.525755641020693 }, { "x": 1451, "y": -1.7113056729056417
}, { "x": 1452, "y": 6.421507982663972 }, { "x": 1453, "y": -
3.0004887069027273 }, { "x": 1454, "y": -8.295303191807925 }, { "x": 1455,
"y": 5.059756843027035 }, { "x": 1456, "y": 8.181025511560343 }, { "x":
1457, "y": -2.0675726283043865 }, { "x": 1458, "y": -4.8230437982624546 }, {
"x": 1459, "y": 8.432821540726291 }, { "x": 1460, "y": -7.616579092737712 },
{ "x": 1461, "y": 2.2721277220792686 }, { "x": 1462, "y": -8.903093509824096
}, { "x": 1463, "y": -2.1736155139867543 }, { "x": 1464, "y":
5.273224352712731 }, { "x": 1465, "y": -1.1777026858590123 }, { "x": 1466,
"y": 7.936412573240492 }, { "x": 1467, "y": -2.1570174794492747 }, { "x":
1468, "y": -5.521034083595325 }, { "x": 1469, "y": -8.720875752785958 }, {
"x": 1470, "y": 5.984192643670989 }, { "x": 1471, "y": 0.9736319082565128 },
{ "x": 1472, "y": -2.1131381963080074 }, { "x": 1473, "y": -
6.990498927845324 }, { "x": 1474, "y": -7.714607309034534 }, { "x": 1475,
"y": 7.260095885165924 }, { "x": 1476, "y": -5.885582294569783 }, { "x":
1477, "y": -1.8240294259429932 }, { "x": 1478, "y": -7.510712300627764 }, {
"x": 1479, "y": 6.085727649848124 }, { "x": 1480, "y": 0.4196053678056355 },
{ "x": 1481, "y": -7.145082337815168 }, { "x": 1482, "y": 8.036407676338019
}, { "x": 1483, "y": 1.40474685499602 }, { "x": 1484, "y":
2.5436713569699823 }, { "x": 1485, "y": 6.624993854592569 }, { "x": 1486,
"y": 4.362610315027583 }, { "x": 1487, "y": -3.3067663394417046 }, { "x":
1488, "y": 3.7461069164443153 }, { "x": 1489, "y": 7.170958759614436 }, {
"x": 1490, "y": 8.600959541439202 }, { "x": 1491, "y": -6.339563594866334 },
{ "x": 1492, "y": 7.953812274598167 }, { "x": 1493, "y": 5.2187035292892645
}, { "x": 1494, "y": -8.161830731442839 }, { "x": 1495, "y": -
1.3038225007313269 }, { "x": 1496, "y": -6.441262160578672 }, { "x": 1497,
"y": 1.5280520338892796 }, { "x": 1498, "y": 8.831744594875214 }, { "x":
1499, "y": 1.5938393947494731 }, { "x": 1500, "y": -8.281079194240919 }, {
"x": 1501, "y": 8.752439357538254 }];

var interval;
var chart = new ej.charts.Chart({
    primaryXAxis: { title: 'Time (s)', majorGridLines: { width: 0 } },

```

```

    primaryYAxis: { title: 'Velocity (m/s)', majorGridLines: { width: 0
}, minimum: -15, maximum: 15, interval: 5 },
    series: [
        {
            type: 'Line', xName: 'x', yName: 'y', dataSource: downdata,
            animation: { enable: false }, width: 2
        }
    ],
    chartArea: {
        border: {
            width: 0
        }
    },
    zoomSettings: {
        enableMouseWheelZooming: true,
        enablePinchZooming: true,
        enableSelectionZooming: true,
        mode: 'X',
        enableScrollbar: false
    },
    title: 'Indonesia - Seismograph Analysis',
    tooltip: { enable: false },
    width: '800',
    load: function (args) {
        var threshold = parseInt(args.chart.width) / 8,
            xName = args.chart.series[0].xName,
            yName = args.chart.series[0].yName,
            sampledData = largestTriangleThreeBucket(downdata, threshold,
xName, yName);
        args.chart.series[0].dataSource = sampledData;
    },
    zoomComplete: function(args) {
        var zoomComplete = true;
        if(args.chart.primaryXAxis.zoomFactor<=0.4)
        {
            args.chart.series[0].dataSource =downdata;
        }
        else{
            var threshold = parseInt(chart.width) / 8,
                xName = args.chart.series[0].xName,
                yName = args.chart.series[0].yName,
                sampledData = largestTriangleThreeBucket(downdata, threshold,
xName, yName);
            args.chart.series[0].dataSource = sampledData;
        }
    }
}, '#element');
function largestTriangleThreeBucket(data, threshold, xProperty,
yProperty) {
    yProperty = yProperty || 0;
    xProperty = xProperty || 1;
    var m = Math.floor,
        y = Math.abs,
        f = data.length;
    if (threshold >= f || 0 === threshold) {
        return data;
    }
}

```

```

var n = [],
    t = 0,
    p = (f - 2) / (threshold - 2),
    c = 0,
    v,
    u,
    w;
n[t++] = data[c];
for (var e = 0; e < threshold - 2; e++) {
    for (var g = 0,
        h = 0,
        a = m((e + 1) * p) + 1,
        d = m((e + 2) * p) + 1,
        d = d < f ? d : f,
        k = d - a; a < d; a++) {
        g += +data[a][xProperty], h += +data[a][yProperty];
    }
    for (var g = g / k,
        h = h / k,
        a = m((e + 0) * p) + 1,
        d = m((e + 1) * p) + 1,
        k = +data[c][xProperty],
        x = +data[c][yProperty],
        c = -1; a < d; a++) {
        "undefined" != typeof data[a] &&
        (u = .5 * y((k - g) * (data[a][yProperty] - x) - (k -
data[a][xProperty]) * (h - x)),
        u > c && (c = u, v = data[a], w = a));
    }
    n[t++] = v;
    c = w;
}
n[t++] = data[f - 1];
return n;
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>

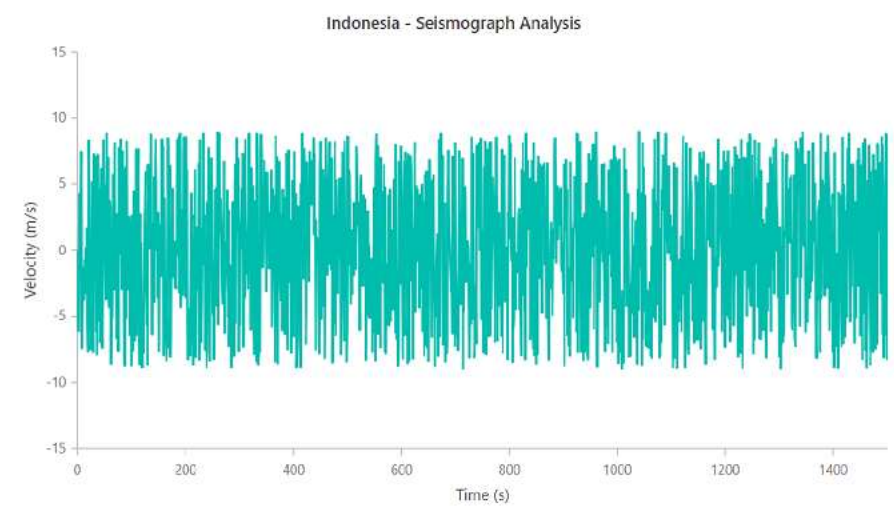
```

```

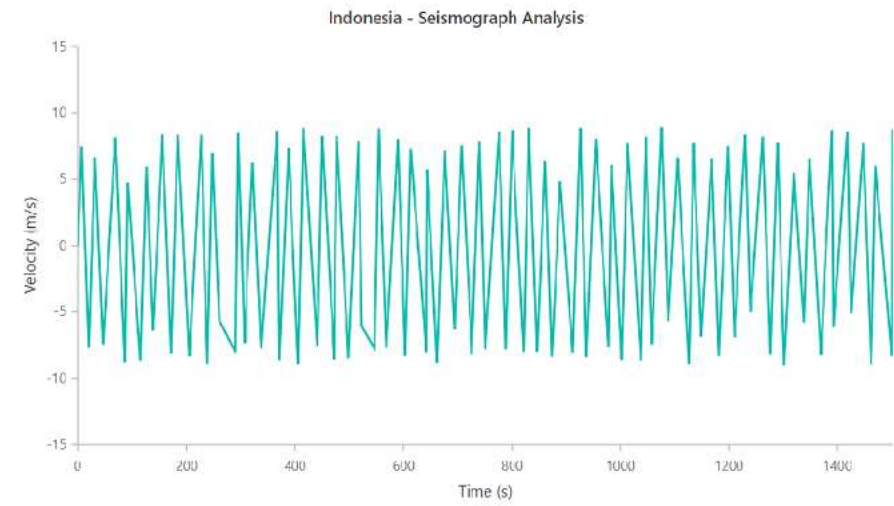
        <div id="element1"></div>
    </div>
    <script>
    var ele = document.getElementById('container');
    if(ele) {
        ele.style.visibility = "visible";
    }
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

Before applying downsampling algorithm



After applying downsampling algorithm



Clicked data in `##Platform_Name##` Chart control
By using the [pointClick](#) event, you can get the chart data of clicked area.
To show the clicked area data from pie, follow the given steps:

Step 1:

By using the [pointClick](#) event, you can get the `args.point.x` and `args.point.y` values.

INDEX.JS

```
var chart = new ej.charts.AccumulationChart({
    // Initialize the chart series
    series: [
        {
            dataSource: [
                { 'x': 'Chrome', y: 37, text: '37%' }, { 'x': 'UC
Browser', y: 17, text: '17%' },
                { 'x': 'iPhone', y: 19, text: '19%' },
                { 'x': 'Others', y: 4, text: '4%' }, { 'x': 'Opera', y:
11, text: '11%' },
                { 'x': 'Android', y: 12, text: '12%' }
            ],
            dataLabel: {
                visible: true, position: 'Inside', name: 'text', font: {
fontWeight: '600' }
            },
            radius: '70%', xName: 'x', yName: 'y', startAngle: 0,
            endAngle: 360, innerRadius: '0%',
            explode: false, explodeOffset: '10%', name: 'Browser',
            animation: {enable: false}
        }
    ],
    enableAnimation: false,
    legendSettings: { visible: false },
    title: 'Mobile Browser Statistics',
    pointClick: function(args){
        document.getElementById("lbl").innerText = "X : " + args.point.x +
"\nY : " + args.point.y;
    }
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
```

```

<body>

  <div class="col-sm-8">
    <div class="row">
      <div class="col-sm-4">
        <div id="container">
          <div id="element" style="width:350px;
height:350px;float:left">
          </div>
          <label id="lbl"></label>
        </div>
      </div>
      <div class="col-sm-4" style="width:200px;
height:350px;float: right">
        <div id="Grid">
          </div>
        </div>
      </div>
    </div>
  </div>

<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Initial scrollbar in ##Platform_Name## Chart control

By setting `zoomFactor` in `primaryXAxis` and `isZoomed` value as `true` in `load` event and `enableScrollbar` value as `true` in `zoomSettings`, you can make the scrollbar visible in initial rendering of chart.

INDEX.JS

```

var chart = new ej.charts.Chart({
  primaryXAxis: {
    zoomFactor: 0.3,
    valueType: 'DateTime',
    labelFormat: 'y',
    intervalType: 'Years',
    edgeLabelPlacement: 'Shift'
  },
  primaryYAxis: {
    labelFormat: '{value}%',
    rangePadding: 'None',
    minimum: 0,
    maximum: 100,
    interval: 20,
  },
  series: [
    {
      type: 'Line',
      dataSource: [
        { x: new Date(2005, 0, 1), y: 21 },

```



```

        { x: new Date(2006, 0, 1), y: 24 },
        { x: new Date(2007, 0, 1), y: 36 },
        { x: new Date(2008, 0, 1), y: 38 },
        { x: new Date(2009, 0, 1), y: 54 },
        { x: new Date(2010, 0, 1), y: 21 },
        { x: new Date(2011, 0, 1), y: 24 },
        { x: new Date(2012, 0, 1), y: 36 },
        { x: new Date(2013, 0, 1), y: 38 },
        { x: new Date(2014, 0, 1), y: 54 },
        { x: new Date(2015, 0, 1), y: 21 },
        { x: new Date(2016, 0, 1), y: 24 },
        { x: new Date(2017, 0, 1), y: 36 },
        { x: new Date(2018, 0, 1), y: 38 },
    ],
    xName: 'x', width: 2, marker: {
        visible: true,
        width: 10,
        height: 10
    },
    yName: 'y', name: 'Germany',
},
],
zoomSettings:
{
    enableMouseWheelZooming: true,
    enablePinchZooming: true,
    enableSelectionZooming: true,
    mode: 'X',
    enableScrollbar: true,
},
title: 'Inflation - Consumer Price',
tooltip: {
    enable: true
},
load: function(args) {
    args.chart.zoomModule.isZoomed = true;
}
}, '#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet">
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div class="col-sm-8">
        <div class="row">
            <div class="col-sm-4">
                <div id="container">
                    <div id="element" style="width:350px;
height:350px;float:left">
                </div>
                <label id="lbl"></label>
            </div>
            <div class="col-sm-4" style="width:200px;
height:350px;float: right">
                <div id="Grid">
                </div>
            </div>
        </div>
    </div>

<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Legend in table in ##Platform_Name## Chart control

The **annotation** property is used to add legend in table and the **multiLevelLabels** property is used to customize the axis label in table format.

To add legend in table with x-axis labels, follow the given steps:

Step 1:

Initialize the custom elements using the **annotation** property.

Create table and rectangle shapes in the html page and set this to the **content** property of annotation.

By setting **coordinateUnits** value to **pixel** in annotation object, you can place the legend in chart based on pixel values.

INDEX.JS

```

var columnData = [
    { x: 'MWBE', y: 2800000, y1: 1000000 },
    { x: 'LDB', y: 1000000, y1: 1550000 },
    { x: 'VBE', y: 2200000, y1: 1200000 },
    { x: 'DBE', y: 3000000, y1: 1000000 },
    { x: 'MWBE', y: 1000000, y1: 800000 },
    { x: 'LDB', y: 500000, y1: 400000 },
    { x: 'VBE', y: 2100000, y1: 1500000 },

```

```

    { x: 'DBE', y: 900000, y1: 0 }
];
var firstClick = true;
var chart = new ej.charts.Chart({
  primaryXAxis: {
    valueType: 'Category',
    border: { width: 1, type: 'Rectangle' },
    isIndexed: true,
    interval: 1,
    majorGridLines: { width: 0 },
    multiLevelLabels: [
      {
        border: { type: 'Rectangle' },
        textStyle: { fontWeight: 'Bold' },
        categories: [
          { start: -0.5, end: 3.5, text: 'Construction', },
          { start: 3.5, end: 7.5, text: 'Professional Services',
        },
      ]
    ],
  },
  {
    border: { type: 'Rectangle' },
    categories: [
      { start: -0.5, end: 0.5, text: '248,952,090' },
      { start: 0.5, end: 1.5, text: '248,952,090' },
      { start: 1.5, end: 2.5, text: '248,952,090' },
      { start: 2.5, end: 3.5, text: '248,952,090' },
      { start: 3.5, end: 4.5, text: '248,952,090' },
      { start: 4.5, end: 5.5, text: '248,952,090' },
      { start: 5.5, end: 6.5, text: '248,952,090' },
      { start: 6.5, end: 7.5, text: '248,952,090' },
    ]
  },
  {
    border: { type: 'Rectangle' },
    categories: [
      { start: -0.5, end: 0.5, text: '248,952,090' },
      { start: 0.5, end: 1.5, text: '248,952,090' },
      { start: 1.5, end: 2.5, text: '248,952,090' },
      { start: 2.5, end: 3.5, text: '248,952,090' },
      { start: 3.5, end: 4.5, text: '248,952,090' },
      { start: 4.5, end: 5.5, text: '248,952,090' },
      { start: 5.5, end: 6.5, text: '248,952,090' },
      { start: 6.5, end: 7.5, text: '248,952,090' },
    ]
  },
],
  annotations: [
    {
      content: '#templateWrap',
      coordinateUnits: 'Pixel',
      x: 120,
      y: 328
    },
    {
      content: '#templateWrap1',

```

```

        coordinateUnits: 'Pixel',
        x: 116,
        y: 300
    },
    {
        content: '#templateWrap2',
        coordinateUnits: 'Pixel',
        x: 116,
        y: 328
    },
],
margin: { left: 100, right: 100 },
primaryYAxis:
{
    labelFormat: '{value}%',
},
//Initializing Chart Series
series: [
    {
        type: 'Column', xName: 'x', width: 2, yName: 'y',
        dataSource: columnData, fill: 'skyblue',
        marker: { dataLabel: { visible: true, position: 'Top', font: {
fontWeight: '600', color: 'ffffff' } } }
    },
    {
        type: 'Column', xName: 'x', width: 2, yName: 'y1', fill:
'#b22222',
        dataSource: columnData,
        marker: { dataLabel: { visible: true, position: 'Top', font: {
fontWeight: '600', color: 'ffffff' } } }
    },
],
tooltip: { enable: true },
chartMouseClick: (args) => {
    if ((args.x >= 83 && args.x <= 155) && (args.y > 312 && args.y <
343)) {
        chart.series[0].dataSource = firstClick ? null : columnData;
        chart.refresh();
        firstClick = !firstClick;
    }
},
axisLabelRender: (args) => {
    if (args.axis.name === 'primaryXAxis') {
        args.text = args.text.split(',')[0];
    }
},
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
<title>EJ2 Animation</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">

```

```

<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
    <script id="templateWrap" type="text/x-template">
        <div style='width:80px; padding: 5px;'>
            <table style='width: 100% '>
                <tr>
                    <td><div id='first' style='width:
10px; height: 10px; background:skyblue;'></div></td>
                    <td style='padding-left:
5px;'>Awarded</td>
                </tr>
                <tr>
                    <td><div id='second' style='width:
10px; height: 10px; background:#b22222;'></div></td>
                    <td style='padding-left:
5px;'>Paid</td>
                </tr>
            </table>
        </div>
    </script>
    <script id="templateWrap1" type="text/x-template">
        <div style='width:80px; padding: 5px;'>
            <table style='width: 100% '>
                <tr>
                    <td><div style='border-style:
solid;border-width: 1px; height:28px;width:85px; border-color:
#b5b5b5;'></div></td>
                </tr>
            </table>
        </div>
    </script>
    <script id="templateWrap2" type="text/x-template">
        <div style='width:80px; padding: 5px;'>
            <table style='width: 100% '>
                <tr>
                    <td><div style='border-style:
solid;border-width: 1px; height:28px;width:85px; border-color:
#b5b5b5;'></div></td>
                </tr>
            </table>
        </div>
    </script>
</script>

```

```

var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Syn pan in ##Platform_Name## Chart control

Using the [chartMouseMove](#) event, you can achieve the synchronized panning between multiple charts.

To make a synchronized panning chart, follow the given steps:

Step 1:

Initially create two charts, and enable `zoomSettings` for both charts.

To use the [chartMouseMove](#) event, assign the first chart's `zoomFactor` and `zoomPosition` values to the second chart. Now, pan the first zoomed chart, and then the second chart will be panned automatically based on `zoomFactor` and `zoomPosition`.

The following code sample demonstrates the output.

INDEX.JS

```

var columnData = [{ country: "USA", gold: 50 }, { country: "China", gold: 40 }, { country: "Japan", gold: 70 }, { country: "Australia", gold: 60 }, { country: "France", gold: 50 }, { country: "Germany", gold: 40 }, { country: "Italy", gold: 40 }, { country: "Sweden", gold: 30 }];

var chart = new ej.charts.Chart({
    primaryXAxis: {
        valueType: 'Category',
    },
    primaryYAxis: {
        title: 'Medals'
    },
    annotations: [{
        content: '<div id ="test" style="border-top:3px dashed grey;border-top-width: 2px; width: 10000px"></div>',
        x: 'France',
        y: 50,
        coordinateUnits: 'Point',
        Region: 'Chart'
    }],
    series: [{
        dataSource: columnData,
        xName: 'country', yName: 'gold',
        type: 'Line'
    }],
}, '#element');

```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
```

```

<title>EJ2 Animation</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
        <div id="element1"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Axis hide in ##Platform_Name## Chart control

By using the [chartMouseClicked](#) event, you can hide the axis line through legend.

To hide the axis line through legend click, follow the given steps:

Step 1:

Create a chart with multiple axes.

By using the [chartMouseClicked](#) event, you can get the legend's target ids. Using this event, you can also get the `yAxisName` of each axis, based on which you can hide the axis line when clicking the legend.

The following code sample demonstrates the output.

INDEX.JS

```

var chartData = [
    { x: new Date(1975, 0, 1), y: 16, y1: 10, y2: 4.5 },
    { x: new Date(1980, 0, 1), y: 12.5, y1: 7.5, y2: 5 },
    { x: new Date(1985, 0, 1), y: 19, y1: 11, y2: 6.5 },
    { x: new Date(1990, 0, 1), y: 14.4, y1: 7, y2: 4.4 },
    { x: new Date(1995, 0, 1), y: 11.5, y1: 8, y2: 5 },
    { x: new Date(2000, 0, 1), y: 14, y1: 6, y2: 1.5 },
    { x: new Date(2005, 0, 1), y: 10, y1: 3.5, y2: 2.5 },
    { x: new Date(2010, 0, 1), y: 16, y1: 7, y2: 3.7 }
];
var chart = new ej.charts.Chart({
    primaryXAxis: {
        title: 'Years',

```

```

        lineStyle: { width: 0 },
        labelFormat: 'y',
        intervalType: 'Years',
        valueType: 'DateTime',
        edgeLabelPlacement: 'Shift'
    },
    primaryYAxis:
    {
        title: 'Percentage (%)',
        minimum: 0, maximum: 20, interval: 2,
        labelFormat: '{value}%'
    },
    series: [
        {
            type: 'StepLine',
            dataSource: chartData, xName: 'x', yName: 'y',
            width: 2, name: 'China',
            marker: {
                visible: true, width: 10, height: 10
            },
        },
        {
            type: 'StepLine',
            dataSource: chartData, xName: 'x', yName: 'y1',
            width: 2, name: 'Australia',
            marker: {
                visible: true, width: 10, height: 10
            },
        },
        {
            type: 'StepLine',
            dataSource: chartData, xName: 'x', yName: 'y2',
            width: 2, name: 'Japan',
            marker: {
                visible: true, width: 10, height: 10
            },
        },
    ],
    //Title for chart
    title: 'Unemployment Rates 1975-2010',
    titleStyle: {
        fontFamily: 'Arial',
        fontStyle: 'italic',
        fontWeight: 'regular',
        color: '#E27F2D',
        size: '23px'
    },
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">

```



```

<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
        <div id="element1"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Exact date in ##Platform_Name## Chart control

By using button click you can show exact time or month after scrolling. Here we have changed **zoomFactor** and **zoomPosition** based on time or month requirement. After scrolling you want show full month or full day data then you will customize the zoomFactor and zoomPosition of the chart and show exact time or month.

INDEX.JS

```


```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="lastdata">Last Month</button>
    </div>

```

```

        <button id="chartmode">Switch To Hour</button>
        <div id="element" style="height: 300px"></div>
    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Dialog chart in ##Platform_Name## Chart control

Using the `content` property of the dialog component, you can show the chart in dialog pop-up.

To show the chart in dialog component, follow the given steps:

Step 1:

Initialize the dialog and button components, and then create a basic chart and set the visibility of dialog to `false` when initialize.

By setting the chart `id` in the `content` property of dialog component, you can show chart when clicking the button component.

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="target">
        <div id="container" align="center"></div>
    </div>
    <div id="defaultDialog">
        <div id="container2" align="center"></div>
    </div>
    <button class="e-control e-btn" id="targetButton" role="button" e-
    ripple="true">Open Dialog</button>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>

```

```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Series visible in ##Platform_Name## Chart control

By using the `chartMouseClicked` event, you can show the series based on respective legend click. In this event, you can get the legend target id, using which you can get the current series index. Based on the index, you can set value of `visible` to `true` or `false`.

INDEX.JS

```
var columnData = [{ country: "USA", gold: 50 }, { country: "China", gold: 40 }, { country: "Japan", gold: 70 }, { country: "Australia", gold: 60 }, { country: "France", gold: 50 }, { country: "Germany", gold: 40 }, { country: "Italy", gold: 40 }, { country: "Sweden", gold: 30 }];

var chart = new ej.charts.Chart({
  primaryXAxis: {
    valueType: 'Category',
  },
  primaryYAxis: {
    title: 'Medals'
  },
  annotations: [{
    content: '<div id = "test" style="border-top:3px dashed grey;border-top-width: 2px; width: 10000px"></div>',
    x: 'France',
    y: 50,
    coordinateUnits: 'Point',
    Region: 'Chart'
  }],
  series: [{
    dataSource: columnData,
    xName: 'country', yName: 'gold',
    type: 'Line'
  }],
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>
```

```

    <div id="container">
        <div id="element"></div>
        <div id="element1"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Dynamic chart in ##Platform_Name## Chart control

By using html button, you can add the chart dynamically when click the button.

To add the chart dynamically through button click, follow the given steps:

Step 1:

Initially create the html button.

Then create chart inside of button `onClick` function. Now click the button charts will render based on click count.

The following code sample demonstrates the output.

INDEX.JS

```

let count = 0;
document.getElementById('btn').onclick = () => {
    //Create div element dynamically and append to DOM
    var chartEle = document.createElement('div');
    chartEle.id = 'chartContainer' + count;
    document.getElementsByTagName('body')[0].appendChild(chartEle);
    //Created chart here
    var chart = new ej.charts.Chart({
        series: [{
            type: 'Line', xName: 'x', width: 2, marker: { visible: true },
            yName: 'y', name: 'Germany',
            dataSource: [{ x: 1, y: 21 }, { x: 2, y: 24 }, { x: 3, y: 36 },
                { x: 4, y: 38 }, { x: 5, y: 54 }, { x: 6, y: 57 }, { x: 7, y: 70 }
        ]],
        title: 'Inflation - Consumer Price', tooltip: { enable: true },
        height: '400', width: '800'
    });
    chart.appendTo('#' + chartEle.id);
    count++;
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">

```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
    <button id="btn">Add Chart</button>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

CheckBox

Label and size in ##Platform_Name## Check box control

This section explains the different sizes and labels.

Label

The CheckBox caption can be defined by using the [label](#) property. This reduces the manual addition of label for CheckBox. You can customize the label position before or after the CheckBox through the [labelPosition](#) property.

INDEX.JS

```

ej.base.enableRipple(true);
//Label position - Left.
var checkbox = new ej.buttons.CheckBox({ label: 'Left Side Label',
labelPosition: 'Before' });
checkbox.appendTo('#checkbox1');
//Label position - Right.
checkbox = new ej.buttons.CheckBox({ label: 'Right Side Label', checked:
true });
checkbox.appendTo('#checkbox2');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
<title>EJ2 CheckBox</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">

```

```

<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <ul>
            <li><input type="checkbox" id="checkbox1"></li>
            <li><input type="checkbox" id="checkbox2"></li>
        </ul>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    height: 40px;
    width: 30%;
    position: absolute;
    top: 45%;
    left: 45%;
}
.e-checkbox-wrapper {
    margin-top: 18px;
}
li {
    list-style: none;
}

```

Size

The different CheckBox sizes available are default and small. To reduce the size of default CheckBox to small, set the [cssClass](#) property to `e-small`.

INDEX.JS

```
ej.base.enableRipple(true);
//Small CheckBox.
var checkbox = new ej.buttons.CheckBox({ label: 'Small', cssClass: 'e-small'
});
checkbox.appendTo('#checkbox1');
//Default CheckBox.
checkbox = new ej.buttons.CheckBox({ label: 'Default' });
checkbox.appendTo('#checkbox2');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 CheckBox</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <ul>
      <li><input type="checkbox" id="checkbox1"></li>
      <li><input type="checkbox" id="checkbox2"></li>
    </ul>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

STYLES.CSS

```
#container {
  visibility: hidden;
}
#loader {
  color: #008cff;
  height: 40px;
```

```
width: 30%;
position: absolute;
top: 45%;
left: 45%;
}
.e-checkbox-wrapper {
margin-top: 18px;
}
li {
list-style: none;
}
```

See Also

- [CheckBox customization](#)


Accessibility in ##Platform_Name## Check box control


The Check box component followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.


The accessibility compliance for the Check box component is outlined below.


| Accessibility Criteria | Compatibility |


| -- | -- |


| [WCAG 2.2](#) Support |  |

| [Section 508](#) Support |  |


| Screen Reader Support |  |


| Right-To-Left Support |  |

| Color Contrast |  |

| Mobile Device Support |  |

| Keyboard Navigation Support |  |

| [Accessibility Checker](#) Validation |  |

| [Axe-core](#) Accessibility Validation |  |

<style>

.post .post-content img {


```
display: inline-block;
margin: 0.5em 0;
}
</style>

<div> - All
features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The
component does not meet the requirement.</div>
```

WAI-ARIA attributes

The Check box component followed the [WAI-ARIA](https://www.w3.org/WAI/ARIA/apg/patterns/Check box/) patterns to meet the accessibility. The following ARIA attributes are used in the Check box component:

Attributes	Purpose
---	---
aria-disabled	Indicates that the element is perceivable but disabled, so it is not editable or otherwise operable.

Keyboard interaction

The Check box component followed the [keyboard interaction](https://www.w3.org/WAI/ARIA/apg/patterns/Check box/#keyboardinteraction) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Check box component.

Press	To do this
---	---
Space	When the Check box has focus, pressing the Space key changes the state of the Check box.

Ensuring accessibility

The Check box component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Check box component is shown in the following sample. Open the [sample](https://ej2.syncfusion.com/accessibility/Check box.html) in a new window to evaluate the accessibility of the Check box component with accessibility tools.

See also

- [Accessibility in Syncfusion ##Platform_Name## components](#)

How To

Customized checkbox in ##Platform_Name## Check box control

Customize CheckBox Appearance

You can customize the appearance of the CheckBox component using the CSS rules. Define own CSS rules according to your requirement and assign the class name to the [cssClass](#) property.

The background and border color of the CheckBox is customized through the custom classes to create primary, success, warning, and danger info type of checkbox.

INDEX.JS

```
// To customize CheckBox appearance
// Refer the 'e-primary' class details in 'style.css'.
var checkbox = new ej.buttons.CheckBox({ label: 'Primary', cssClass: 'e-
primary', checked: true });
checkboxbox.appendTo('#checkbox1');
// Refer the 'e-success' class details in 'style.css'.
checkboxbox = new ej.buttons.CheckBox({ label: 'Success', cssClass: 'e-
success', checked: true });
checkboxbox.appendTo('#checkbox2');
// Refer the 'e-info' class details in 'style.css'.
checkboxbox = new ej.buttons.CheckBox({ label: 'Info', cssClass: 'e-info',
checked: true });
checkboxbox.appendTo('#checkbox3');
// Refer the 'e-warning' class details in 'style.css'.
checkboxbox = new ej.buttons.CheckBox({ label: 'Warning', cssClass: 'e-
warning', checked: true });
checkboxbox.appendTo('#checkbox4');
// Refer the 'e-danger' class details in 'style.css'.
checkboxbox = new ej.buttons.CheckBox({ label: 'Danger', cssClass: 'e-danger',
checked: true });
checkboxbox.appendTo('#checkbox5');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 CheckBox</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
```

```

<div id="container">
  <ul>
    <li><input type="checkbox" id="checkbox1"></li>
    <li><input type="checkbox" id="checkbox2"></li>
    <li><input type="checkbox" id="checkbox3"></li>
    <li><input type="checkbox" id="checkbox4"></li>
    <li><input type="checkbox" id="checkbox5"></li>
  </ul>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
  visibility: hidden;
}
#loader {
  color: #008cff;
  height: 40px;
  width: 30%;
  position: absolute;
  top: 45%;
  left: 45%;
}
li {
  list-style: none;
}
.e-checkbox-wrapper {
  margin-top: 18px;
}
.e-checkbox-wrapper.e-primary:hover .e-frame.e-check { /* csslint allow:
adjoining-classes */
  background-color: #e03872;
}
.e-checkbox-wrapper.e-success .e-frame.e-check,
.e-checkbox-wrapper.e-success .e-checkbox:focus + .e-frame.e-check { /*
csslint allow: adjoining-classes */
  background-color: #689f38;
}
.e-checkbox-wrapper.e-success:hover .e-frame.e-check { /* csslint allow:
adjoining-classes */
  background-color: #449d44;
}
.e-checkbox-wrapper.e-info .e-frame.e-check,
.e-checkbox-wrapper.e-info .e-checkbox:focus + .e-frame.e-check { /* csslint
allow: adjoining-classes */
  background-color: #2196f3;
}

```

```
.e-checkbox-wrapper.e-info:hover .e-frame.e-check { /* csslint allow:
adjoining-classes */
  background-color: #0b7dda;
}
.e-checkbox-wrapper.e-warning .e-frame.e-check,
.e-checkbox-wrapper.e-warning .e-checkbox:focus + .e-frame.e-check { /*
csslint allow: adjoining-classes */
  background-color: #ef6c00;
}
.e-checkbox-wrapper.e-warning:hover .e-frame.e-check { /* csslint allow:
adjoining-classes */
  background-color: #cc5c00;
}
.e-checkbox-wrapper.e-danger .e-frame.e-check,
.e-checkbox-wrapper.e-danger .e-checkbox:focus + .e-frame.e-check { /*
csslint allow: adjoining-classes */
  background-color: #d84315;
}
.e-checkbox-wrapper.e-danger:hover .e-frame.e-check { /* csslint allow:
adjoining-classes */
  background-color: #ba3912;
}
```

Custom Frame

CheckBox frame can be customized as per the requirement by adding CSS rules.

In the following example, to-do list is displayed with round checkbox by changing `border-radius` as `100%` by adding `e-custom` class.

INDEX.JS

```
// To customize CheckBox frame appearance
var checkbox = new ej.buttons.CheckBox({ label: 'Buy groceries', cssClass:
'e-custom', checked: true });
checkbox.appendTo('#checkbox1');
checkbox = new ej.buttons.CheckBox({ label: 'Pay rent', cssClass: 'e-custom'
});
checkbox.appendTo('#checkbox2');
checkbox = new ej.buttons.CheckBox({ label: 'Make Dinner', cssClass: 'e-
custom' });
checkbox.appendTo('#checkbox3');
checkbox = new ej.buttons.CheckBox({ label: 'Finish To-do list article',
cssClass: 'e-custom' });
checkbox.appendTo('#checkbox4');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 CheckBox</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <ul>
            <li><input type="checkbox" id="checkbox1"></li>
            <li><input type="checkbox" id="checkbox2"></li>
            <li><input type="checkbox" id="checkbox3"></li>
            <li><input type="checkbox" id="checkbox4"></li>
        </ul>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    height: 40px;
    width: 30%;
    position: absolute;
    top: 45%;
    left: 45%;
}
li {
    list-style: none;
}
.e-checkbox-wrapper {
    margin-top: 18px;
}
.e-custom .e-frame {
    border-radius: 100%;
}
.e-checkicon.e-checkbox-wrapper .e-frame.e-check::before {
    content: '\e77d';
}
.e-checkicon.e-checkbox-wrapper .e-check {
    font-size: 8.5px;
}

```

```

}
.e-checkicon.e-checkbox-wrapper .e-frame.e-check {
  background-color: white;
  border-color: grey;
  color: grey;
}
.e-checkicon.e-checkbox-wrapper:hover .e-frame.e-check {
  background-color: white;
  border-color: grey;
  color: grey;
}
.e-checkicon.e-checkbox-wrapper .e-checkbox:focus + .e-frame.e-check {
  background-color: white;
  border-color: grey;
  box-shadow: none;
  color: grey;
}
}

```

Custom Check Icon

CheckBox check icon can be customized as per the requirement by adding CSS rules.

In the following example, the check icon can be customized by changing check icon content, background and border color in focus and hovered states by adding `e-checkicon` class.

INDEX.JS

```

// To customize CheckBox frame appearance
var checkbox = new ej.buttons.CheckBox({ label: 'Buy groceries', cssClass:
'e-checkicon', checked: true });
checkbox.appendTo('#checkbox1');
checkbox = new ej.buttons.CheckBox({ label: 'Pay rent', cssClass: 'e-
checkicon' });
checkbox.appendTo('#checkbox2');
checkbox = new ej.buttons.CheckBox({ label: 'Make Dinner', cssClass: 'e-
checkicon' });
checkbox.appendTo('#checkbox3');
checkbox = new ej.buttons.CheckBox({ label: 'Finish To-do list article',
cssClass: 'e-checkicon' });
checkbox.appendTo('#checkbox4');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 CheckBox</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <ul>
            <li><input type="checkbox" id="checkbox1"></li>
            <li><input type="checkbox" id="checkbox2"></li>
            <li><input type="checkbox" id="checkbox3"></li>
            <li><input type="checkbox" id="checkbox4"></li>
        </ul>
    </div>
</body>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    height: 40px;
    width: 30%;
    position: absolute;
    top: 45%;
    left: 45%;
}
li {
    list-style: none;
}
@font-face {
    font-family: 'btn-icon';
    src:
        url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAKAIAAAwAgTlMvMjltSfgAAAEoAAAAVmNtYXNlH+dzAAABoAAAAEJnbHlm1v4
8pAAAAfgAAAQYaGVhZBOPfZcAAADQAAAAANmhoZWEIUQQJAAAArAAAACRobXR4IAAAAAAAYAAAAA
gbG9jYQON6ApQAAAHkAAAAEmlheHABFQCqAAABCAAAACBuYW1l07lFxAABhAAAAIxcG9zdK9uovo
AAAhEAAAAGaAABAAAEAAAAAFWEAAAAAAD9AABAAAAAIAAAAAAACAABAAAAAQAAJ1LUzF8
PPPUACwQAAAAAANg+nFMAAAAA2D6cUwAAAAAD9AP0AAAACAACAAAAAIAIAJ4AAwAAAAA
AAgAAAAoACgAAAP8AAAAAQAQAAZAAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnBgQAAAAAXAQAAAAAABAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAABAAAAAQAAAAAACAaaaaaWAAABQAAwABAAAAFAAEAC4AAAAEAAQAAQA
A5wb//wAA5wD//wAAAAEABAAAAEAAgADAAQABQAGAAcAAAAAAdgAkADIAhAEuAewCDAAAAAE
AAAAA2ED9AACAA3CQGeAsT9PAwB9AH0AAACAAAAAAPHa/QAAwAHAAALIREhASERIQJpAV7+ov3
QAV7+ogwD6PwYA+gAAAEAAAAA4sD9AACAAATARF0AxgCAP4MA+gAAAAABAAAAAAP0A/QAQwAAExE
fDyE/DxEvDyEPDgwBAGMFBQCICQkLCwwMDQ4NatoNDg0MDAsLCQkIBwUFAwIBAQIDBQUHCAkJCws

```

```
MDA00Df0mDQ4NDawLCwkJCAcFBQMCA239Jg4NDQ0LCwsJCQgHBQUDAgEBAgMFBQcICQkLCwsNDQ0
OAtOoDQ0NCwsLCQkIBwUFAwIBAQIDBUHCAkJCwsLDQ0NAAIAAAAAA/MDxQADAIwAADczESMBDwM
VFw8METM3HwQ3Fz8KPQEvBT8LLwg3NT8INS8FNT8NNS8JByU/BDUvCyMPAQytrQH5AgoEAQEBArg
hERESEyIJCsgQBiEHNQceOZPbDgUICw0LCQUDBAICBAkGAgEBAQMOBAkIBgcDAwEBAQEDAwMJAjAgE
BAxYLBQQEawMCAGIEBAoBAQEECGcHBgUFBAMDAQEBAQQFBwkFBQUGEf6tDwkEawIBAQMDCgwVawc
GDAsNBwdaAYcB3gEFAwN2HwoELDodGxwaLwkIGwz+igEBHwMBAQECAQEDBgoKDAYICAgFCAkICwU
EBAQFAwYDBwgIDAgHCACGBgYFBQkEAgYCBawJBgUGBwkJCgkICAcLBAIFAwIEBAQFBQcGBwgHBgY
GBgoJCAYCAGeBAQFGMRkaGw0NDA0LIh4xBAQCBAEBAGADAAAAAOKA/MAHABCAJ0AAAEzHwIRDwM
hLwIDNzM/CjUTHwcVIwcVIy8HETcXmZ8KNscxBxEfDjsBHQEfDTMhMz8OES8PIz0BLw4hA0EDBQQ
DAQIEbf5eBQQCAW4RDg0LCQgGBQUDBAFeBAMDawIBAQL7Y0EawQCAgIBAYYKChEQDQsJCACeBAU
CYt8BAQIDBAUFbQcHBwgICQgKjQECAGMEBAUFBgYHBgcIBwGcCAcHBwYGBgUFBAQDAGIBAQEBAgI
DBAQFBQYGBgcHBwgmAQMDawUFBgYHBwgICQkJ/tQCiwMEbf3XAwYEAgIEBgFoAQEDBQYGBwgIBw0
KhQEiAQEBAGMDawTV+94BAQECAwMDBAGyAQECBAYHCAgJCgkQCaQC6/47CQkICQcIBwYGBQQEawI
CUAgHBwcGBgYFBQQEawMBAgIBAwMEBAUFbQcGBwcHCAImCAcHBwYGBgUFBAQDAGIBAdUJCQgICAg
GBwYFBAQDAGEBAAAAAIAAAAAA6cD9AADAawAADchNSElAQcJAScBESNZAO78sgGB/uMuAXkBgDb
+1EwMTZcBCD3+ngFiPf7pAxMAAAAAABIA3gABAAAAAEEEEAAAAABAAAAAABAAgAAQABAAAAA
CAACACQABAAAAAADAAGAEAAABAAAAAEEAGAGAABAAAAAFAAsAIAABAAAAAAGAAGAKwABAAA
AAAAKACwAMwABAAAAAALABIAXwADAAEECQAAAAIAcQADAAEECQABABAAcwADAAEECQACAA4AgwA
DAAEECQADABAAKQADAAEECQAEABAAoQADAAEECQAFABYAsQADAAEECQAGABAAxwADAAEECQAKAFg
AlwADAAEECQALACQBLyBidG4taWNvblJlZ3VsYXJidG4taWNvbmJ0bilpY29uVmVyc2lubiAxLjB
idG4taWNvbkbZvbnQgZ2VuZXJhdGVkIHVzaW5nIFN5bmNmdXNpb24gTWV0cm8gU3RlZGlvd3d3LnN
5bmNmdXNpb24uY29tACAAYgB0AG4ALQBpAGMABwBuAFIAZQBnAHUAbABhAHIAyGBOAG4ALQBpAGM
AbwBuAGIAAdABuAC0AaQBjAG8AbgBWAGUAcgBzAGkAbwBuACAAMQAuADAAYgB0AG4ALQBpAGMABwB
uAEYAbwBuAHQAIAbnAGUAbgBlAHIAyQB0AGUAZAAGAHUAcwBpAG4AZwAgAFMAeQBwAGMAZgB1AHM
AaQBvAG4AIAbnAGUAdABYAG8AIAbTAHQAdQBkAGkAbwB3AHcAdwAuAHMAeQBwAGMAZgB1AHMAaQB
vAG4ALgBjAG8AbQAAAAACAAAAAoooooAAAAAAAAAAAAAAAAAAAAAgBAGEDAQQBBQE
GAQcBCAEJAaptZWRpYS1wbGF5C2l1ZGlhLXBhdXNlDmFycm93aGVhZC1sZWZ0BHN0b3AJbGlrZS0
tLTAXBGNvcHkQLWRvd25sb2FkLTAyLXdmLQAA) format('trueType');
font-weight: normal;
font-style: normal;
}
.e-icons {
font-family: 'btn-icon' !important;
speak: none;
font-size: 55px;
font-style: normal;
font-weight: normal;
font-variant: normal;
text-transform: none;
line-height: 1;
-webkit-font-smoothing: antialiased;
-moz-osx-font-smoothing: grayscale;
}
.e-checkbox-wrapper {
margin-top: 18px;
}
.e-checkicon.e-checkbox-wrapper .e-frame.e-check::before {
content: '\e703';
}
.e-checkicon.e-checkbox-wrapper .e-check {
font-size: 8px;
}
.e-checkicon.e-checkbox-wrapper .e-frame.e-check {
background-color: white;
border-color: grey;
color: grey;
}
```



```
.e-checkicon.e-checkbox-wrapper:hover .e-frame.e-check {
    background-color: white;
    border-color: grey;
    color: grey;
}
.e-checkicon.e-checkbox-wrapper .e-checkbox:focus + .e-frame.e-check {
    background-color: white;
    border-color: grey;
    box-shadow: none;
    color: grey;
}
.e-checkicon.e-checkbox-wrapper .e-ripple-element {
    background: grey;
}
```

Name and value in form submit in ##Platform_Name## Check box control

The [name](#) attribute of the CheckBox is used to group Checkboxes. When the Checkboxes are grouped in form, the checked items [value](#) attribute will post to the server on form submit that can be retrieved through the name. The disabled and unchecked CheckBox value will not be sent to the server on form submit.

In the following code snippet, Cricket and Hockey are in the checked state, Tennis is in [disabled](#) state and Basketball is in unchecked state. Now, the value that is in checked state only be sent on form submit.

INDEX.JS

```
ej.base.enableRipple(true);
//Name and Value attribute in form submit.
var checkbox = new ej.buttons.CheckBox({ name: 'Sport', value: 'Cricket',
label: 'Cricket', checked: true });
checkbox.appendTo('#checkbox1');
checkbox = new ej.buttons.CheckBox({ name: 'Sport', value: 'Hockey', label:
'Hockey', checked: true });
checkbox.appendTo('#checkbox2');
checkbox = new ej.buttons.CheckBox({ name: 'Sport', value: 'Tennis', label:
'Tennis', disabled: true });
checkbox.appendTo('#checkbox3');
checkbox = new ej.buttons.CheckBox({ name: 'Sport', value: 'Basketball',
label: 'Basketball' });
checkbox.appendTo('#checkbox4');
var button = new ej.buttons.Button({ isPrimary: true });
button.appendTo('#btnElement');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 CheckBox</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <form>
            <ul>
                <li><input type="checkbox" id="checkbox1"></li>
                <li><input type="checkbox" id="checkbox2"></li>
                <li><input type="checkbox" id="checkbox3"></li>
                <li><input type="checkbox" id="checkbox4"></li>
                <li><button id="btnElement">Submit</button></li>
            </ul>
        </form>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    height: 40px;
    width: 30%;
    position: absolute;
    top: 45%;
    left: 45%;
}
.e-checkbox-wrapper {
    margin-top: 18px;
}
button {
    margin: 20px 0 0 5px;
}
li {
    list-style: none;
}

```

Right to left in ##Platform_Name## Check box control

CheckBox component has RTL support. This can be achieved by setting `enableRtl` as `true`.

The following example illustrates how to enable right-to-left support in CheckBox component.

INDEX.JS

```
ej.base.enableRipple(true);  
// Initialize CheckBox component.  
var checkbox = new ej.buttons.CheckBox({ label: 'Default', enableRtl: true  
});  
// Render initialized CheckBox.  
checkbox.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>  
  <title>EJ2 CheckBox</title>  
  <meta charset="utf-8">  
  <meta name="viewport" content="width=device-width, initial-scale=1.0">  
  <meta name="description" content="Typescript UI Controls">  
  <meta name="author" content="Syncfusion">  
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-  
base/styles/material.css" rel="stylesheet">  
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-  
buttons/styles/material.css" rel="stylesheet">  
  <link href="styles.css" rel="stylesheet">  
  
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"  
type="text/javascript"></script>  
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type  
="text/javascript"></script>  
</head>  
<body>  
  
  <div id="container">  
    <input type="checkbox" id="element">  
  </div>  
  <script>  
var ele = document.getElementById('container');  
if(ele) {  
  ele.style.visibility = "visible";  
}  
  </script>  
  <script src="index.js" type="text/javascript"></script>  
</body></html>
```

STYLES.CSS

```
#container {  
  visibility: hidden;  
}  
#loader {  
  color: #008cff;  
  height: 40px;
```

```

width: 30%;
position: absolute;
top: 45%;
left: 45%;
}
.e-checkbox-wrapper {
margin-top: 18px;
}

```

Chips

Types in ##Platform_Name## Chips control

The ChipList control has the following types.

- Input Chip
- Choice Chip
- Filter Chip
- Action Chip

Input Chip

Input Chip holds information in compact form. It converts user input into chips.

INDEX.JS

```

new ej.buttons.ChipList({chips: ['Andrew', 'Janet', 'Laura', 'Margaret']},
'#chip');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Chip</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="chip"></div>
  </div>
</script>

```

```

var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
    display: inline-block;
    position: relative;
    left: 50%;
    top: 100px;
    transform: translateX(-50%);
}
#loader {
    color: #008cff;
    height: 40px;
    width: 30%;
    position: absolute;
    font-family: 'Helvetica Neue', 'calibiri';
    font-size: 16px;
    top: 45%;
    left: 45%;
}

```

Choice Chip

Choice Chip allows you to select a single chip from the set of ChipList/ChipCollection. It can be enabled by setting the `selection` property to `Single`.

INDEX.JS

```

new ej.buttons.ChipList({chips: ['Small', 'Medium', 'Large', 'Extra Large'],
selection: "Single"}, '#chip');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Chip</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="styles.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>

```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="chip"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
    display: inline-block;
    position: relative;
    left: 50%;
    top: 100px;
    transform: translateX(-50%);
}
#loader {
    color: #008cff;
    height: 40px;
    width: 30%;
    position: absolute;
    font-family: 'Helvetica Neue', 'calibiri';
    font-size: 16px;
    top: 45%;
    left: 45%;
}

```

Filter Chip

Filter Chip allows you to select a multiple chip from the set of ChipList/ChipCollection. It can be enabled by setting the **selection** property to **Multiple**.

INDEX.JS

```

new ej.buttons.ChipList({chips: ['Chai', 'Chang', 'Aniseed Syrup', 'Ikura'],
selection: "Multiple"}, '#chip');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Chip</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="chip"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
    display: inline-block;
    position: relative;
    left: 50%;
    top: 100px;
    transform: translateX(-50%);
}
#loader {
    color: #008cff;
    height: 40px;
    width: 30%;
    position: absolute;
    font-family: 'Helvetica Neue', 'calibiri';
    font-size: 16px;
    top: 45%;
    left: 45%;
}

```

Action Chip

The Action Chip triggers the event like click or delete, which helps doing action based on the event.

INDEX.JS

```

new ej.buttons.ChipList({ chips: ['Send a text', 'Set a remainder', 'Read my
emails ', 'Set alarm'],
    click: function(e) {
        alert('you have clicked ' + e.text)
    }
});

```

```
}}, '#chip');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Chip</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="chip"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

STYLES.CSS

```
#container {
  visibility: hidden;
  display: inline-block;
  position: relative;
  left: 50%;
  top: 100px;
  transform: translateX(-50%);
}
#loader {
  color: #008cff;
  height: 40px;
  width: 30%;
  position: absolute;
  font-family: 'Helvetica Neue', 'calibiri';
  font-size: 16px;
  top: 45%;
  left: 45%;
}
```


Deletable Chip

Deletable Chip allows you to delete a chip from ChipList/ChipCollection. It can be enabled by setting the `enableDelete` property to `true`.

INDEX.JS

```
new ej.buttons.ChipList({chips: ['Send a text', 'Set a remainder', 'Read my emails', 'Set alarm'], enableDelete: true}, '#chip');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Chip</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="chip"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

STYLES.CSS

```
#container {
  visibility: hidden;
  display: inline-block;
  position: relative;
  left: 50%;
  top: 100px;
  transform: translateX(-50%);
}
#loader {
```

```
color: #008cff;
height: 40px;
width: 30%;
position: absolute;
font-family: 'Helvetica Neue','calibiri';
font-size:16px;
top: 45%;
left: 45%;
}
```

Customization in ##Platform_Name## Chips control

This section explains the customization of styles, leading icons, avatar, and trailing icons in Chip control.

Styles

The Chip control has the following predefined styles that can be defined using the `cssClass` property.

Class	Description
-----	-----
e-primary	Represents a primary chip.
e-success	Represents a positive chip.
e-info	Represents an informative chip.
e-warning	Represents a chip with caution.
e-danger	Represents a negative chip.

INDEX.JS

```
new ej.buttons.ChipList({ chips: [{
  "text": "Apple",
  "cssClass": "e-primary"
},
{
  "text": "Microsoft",
  "cssClass": "e-info"
},
{
  "text": "Google",
  "cssClass": "e-success"
},
{
  "text": "Tesla",
  "cssClass": "e-warning"
},
{
  "text": "Intel",
  "cssClass": "e-danger"
}
] }, '#chip');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
```

```

<title>EJ2 Chip</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="chip"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#loader {
    color: #008cff;
    height: 40px;
    width: 30%;
    position: absolute;
    font-family: 'Helvetica Neue','calibiri';
    font-size:16px;
    top: 45%;
    left: 45%;
}
#chip .andrew {
background-image:
url('https://ej2.syncfusion.com/demos/src/chips/images/andrew.png')
}
#chip .margaret {
background-image:
url('https://ej2.syncfusion.com/demos/src/chips/images/margaret.png')
}
#chip .laura {
background-image:
url('https://ej2.syncfusion.com/demos/src/chips/images/laura.png')
}
#chip .janet {

```

```
background-image:
url('https://ej2.syncfusion.com/demos/src/chips/images/janet.png')
}
```

Leading Icon

You can add and customize the leading icon of chip using the `leadingIconCss` property.

INDEX.JS

```
new ej.buttons.ChipList({ chips: [{
  "text": "Anne",
  "leadingIconCss": "andrew"
},
{
  "text": "Janet",
  "leadingIconCss": "janet"
},
{
  "text": "Laura",
  "leadingIconCss": "laura"
},
{
  "text": "Margaret",
  "leadingIconCss": "margaret"
}
]}, '#chip');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Chip</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="chip"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
```

```

    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#loader {
  color: #008cff;
  height: 40px;
  width: 30%;
  position: absolute;
  font-family: 'Helvetica Neue','calibiri';
  font-size:16px;
  top: 45%;
  left: 45%;
}
#chip .andrew {
background-image:
url('https://ej2.syncfusion.com/demos/src/chips/images/andrew.png')
}
#chip .margaret {
background-image:
url('https://ej2.syncfusion.com/demos/src/chips/images/margaret.png')
}
#chip .laura {
background-image:
url('https://ej2.syncfusion.com/demos/src/chips/images/laura.png')
}
#chip .janet {
background-image:
url('https://ej2.syncfusion.com/demos/src/chips/images/janet.png')
}

```

Avatar

You can add and customize the avatar of chip using the `avatarIconCss` property.

INDEX.JS

```

new ej.buttons.ChipList({ chips: [{
  "text": "Anne",
  "avatarIconCss": "andrew"
},
{
  "text": "Janet",
  "avatarIconCss": "janet"
},
{
  "text": "Laura",
  "avatarIconCss": "laura"
},
{
  "text": "Margaret",
  "avatarIconCss": "margaret"
}

```

```

    }
  ]}, '#chip');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Chip</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="chip"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#loader {
  color: #008cff;
  height: 40px;
  width: 30%;
  position: absolute;
  font-family: 'Helvetica Neue','calibiri';
  font-size:16px;
  top: 45%;
  left: 45%;
}
#chip .andrew {
background-image:
url('https://ej2.syncfusion.com/demos/src/chips/images/andrew.png')
}
#chip .margaret {
background-image:
url('https://ej2.syncfusion.com/demos/src/chips/images/margaret.png')
}

```

```
}
#chip .laura {
background-image:
url('https://ej2.syncfusion.com/demos/src/chips/images/laura.png')
}
#chip .janet {
background-image:
url('https://ej2.syncfusion.com/demos/src/chips/images/janet.png')
}
```

Avatar Content

You can add and customize the avatar content of chip using the `avatarText` property.

INDEX.JS

```
new ej.buttons.ChipList({ chips: [{
    "text": "Anne",
    "avatarText": "A"
},
{
    "text": "Janet",
    "avatarText": "J"
},
{
    "text": "Laura",
    "avatarText": "L"
},
{
    "text": "Margaret",
    "avatarText": "M"
}
]}, '#chip');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Chip</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
```

```

    <div id="container">
        <div id="chip"></div>
    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#loader {
    color: #008cff;
    height: 40px;
    width: 30%;
    position: absolute;
    font-family: 'Helvetica Neue','calibiri';
    font-size:16px;
    top: 45%;
    left: 45%;
}
#chip .andrew {
background-image:
url('https://ej2.syncfusion.com/demos/src/chips/images/andrew.png')
}
#chip .margaret {
background-image:
url('https://ej2.syncfusion.com/demos/src/chips/images/margaret.png')
}
#chip .laura {
background-image:
url('https://ej2.syncfusion.com/demos/src/chips/images/laura.png')
}
#chip .janet {
background-image:
url('https://ej2.syncfusion.com/demos/src/chips/images/janet.png')
}

```

Trailing Icon

You can add and customize the trailing icon of chip using the `trailingIconCss` property.

INDEX.JS

```

new ej.buttons.ChipList({ chips: [{
    "text": "Anne",
    "trailingIconCss": "e-dlt-btn"
},
{
    "text": "Janet",
    "trailingIconCss": "e-dlt-btn"
},
{

```



```

        "text": "Laura",
        "trailingIconCss": "e-dlt-btn"
    },
    {
        "text": "Margaret",
        "trailingIconCss": "e-dlt-btn"
    }
  ]}, '#chip');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Chip</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="chip"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#loader {
  color: #008cff;
  height: 40px;
  width: 30%;
  position: absolute;
  font-family: 'Helvetica Neue', 'calibiri';
  font-size: 16px;
  top: 45%;
  left: 45%;
}
#chip .andrew {

```

```
background-image:
url('https://ej2.syncfusion.com/demos/src/chips/images/andrew.png')
}
#chip .margaret {
background-image:
url('https://ej2.syncfusion.com/demos/src/chips/images/margaret.png')
}
#chip .laura {
background-image:
url('https://ej2.syncfusion.com/demos/src/chips/images/laura.png')
}
#chip .janet {
background-image:
url('https://ej2.syncfusion.com/demos/src/chips/images/janet.png')
}
```

Outline Chip

Outline chip has the border with the background transparent. It can be set using the `cssClass` property.

INDEX.JS

```
new ej.buttons.ChipList({ chips: ['Chai', 'Chang', 'Aniseed Syrup ',
'Ikura'],
cssClass: 'e-outline'}, '#chip');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Chip</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="chip"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
```

```
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

STYLES.CSS

```
#loader {
  color: #008cff;
  height: 40px;
  width: 30%;
  position: absolute;
  font-family: 'Helvetica Neue', 'calibiri';
  font-size: 16px;
  top: 45%;
  left: 45%;
}
#chip .andrew {
background-image:
url('https://ej2.syncfusion.com/demos/src/chips/images/andrew.png')
}
#chip .margaret {
background-image:
url('https://ej2.syncfusion.com/demos/src/chips/images/margaret.png')
}
#chip .laura {
background-image:
url('https://ej2.syncfusion.com/demos/src/chips/images/laura.png')
}
#chip .janet {
background-image:
url('https://ej2.syncfusion.com/demos/src/chips/images/janet.png')
}
```

Style in ##Platform_Name## Chips control

The following content provides the exact CSS structure that can be used to modify the control's appearance based on the user preference.

Customizing the chip text

Use the following CSS to customize the chip text properties.

```
.e-chip .e-chip-text {
font-size: 20px;
color: black;
font-weight: normal;
}
,
```

Customizing the chip icon

Use the following CSS to customize the chip icon properties.

```
、  
  
.e-chip .e-icon {  
background-image: url('https://ej2.syncfusion.com/demos/src/chips/images/laura.png');  
opacity: 0.8;  
}  
、
```

Customizing the chip delete button

Use the following CSS to customize the chip delete button.

```
、  
  
.e-chip-list .e-chip .e-chip-delete.e-dlt-btn {  
color: #e3165b;  
font-size: 12px;  
}  
、
```

Customizing the chip outline

Use the following CSS to customize the chip outline.

```
、  
  
.e-chip-list .e-chip.e-outline {  
border-color: #e3165b;  
border-width: 3px;  
}  
、
```

Customizing the chip on selection

Use the following CSS to customize the chip on selection.

```
、  
  
/ To customize single chip on selection /  
.e-chip-list.e-selection .e-chip.e-active {  
background-color: #ffca1c;  
color: #e3165b;  
}  
  
/ To customize multiple chip on selection /  
.e-chip-list .e-chip.e-active {  
background-color: #e3165b;  
color: white;
```

```
}  
`
```

Customizing the chip avatar text

Use the following CSS to customize the chip avatar text properties.

```
`  
  
.e-chip-list .e-chip .e-chip-avatar {  
background-color: #d51a1a;  
color: #fafafa;  
}  
`
```

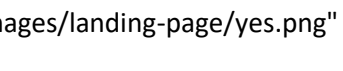
Accessibility in ##Platform_Name## Chips component

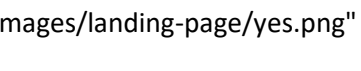
The Chips component followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

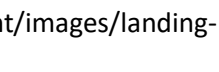
The accessibility compliance for the Chips component is outlined below.

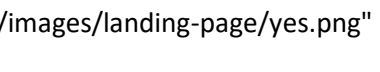
| Accessibility Criteria | Compatibility |

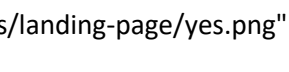
| -- | -- |

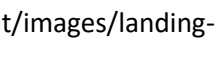
| [WCAG 2.2](#) Support |  |

| [Section 508](#) Support |  |

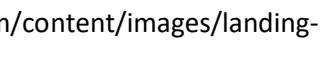
| Screen Reader Support |  |


| Right-To-Left Support |  |

| Color Contrast |  |

| Mobile Device Support |  |

| Keyboard Navigation Support |  |

| [Accessibility Checker](#) Validation |  |

| [Axe-core](#) Accessibility Validation |  |

<style>

```
.post .post-content img {  
display: inline-block;
```

```
margin: 0.5em 0;
```

```
}
```

```
</style>
```

```
<div> - All features of the component meet the requirement.</div>
```

```
<div> - Some features of the component do not meet the requirement.</div>
```

```
<div> - The component does not meet the requirement.</div>
```

WAI-ARIA attributes

The Chips component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Chips component:

Attributes	Purpose
------------	---------

---	---
-----	-----

<code>role=checkbox</code>	Indicates the ChipList component wrapper element as <code>checkbox</code> .
----------------------------	---

<code>role=option</code>	Used to convey a significant and contextual message to the user(ChipList).
--------------------------	--

<code>role=button</code>	Used to convey a significant and contextual message to the user(Single Chip).
--------------------------	---

<code>aria-label</code>	Provides an accessible name for the Chip.
-------------------------	---

<code>aria-selected</code>	Indicates the element is selected.
----------------------------	------------------------------------

<code>aria-disabled</code>	Indicates element is perceivable but disabled.
----------------------------	--

<code>aria-multiselectable</code>	Indicates multiple items to be selected.
-----------------------------------	--

Keyboard interaction

The Chips component followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Chips component.

Keyboard shortcuts	Actions
--------------------	---------

-----	-----
-------	-------

Enter / Space	Selects the targeted chip from the ChipList/ChipCollection.
----------------------	--

Delete / Backspace	Deletes the targeted chip from the ChipList/ChipCollection.
---------------------------	--

Ensuring accessibility

The Chips component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Chips component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Chips component with accessibility tools.

See also

- [Accessibility in Syncfusion ##Platform_Name## components](#)

Circular Gauge

Gauge dimensions in ##Platform_Name## Circular gauge control

Size for Container

Circular gauge can render to its container size. You can set the size via inline or CSS as demonstrated below.

```
,  
  
<div id='container'>  
  <div id='element' style="width:650px; height:350px;"></div>  
</div>  
,
```

INDEX.JS

```
var circulargauge = new ej.circulargauge.CircularGauge({  
  }, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>  
  <title>EJ2 Animation</title>  
  <meta charset="utf-8">  
  <meta name="viewport" content="width=device-width, initial-scale=1.0">  
  <meta name="description" content="Typescript UI Controls">  
  <meta name="author" content="Syncfusion">  
  <link href="index.css" rel="stylesheet">  
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-  
  popups/styles/material.css" rel="stylesheet">  
  
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"  
  type="text/javascript"></script>  
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type  
  ="text/javascript"></script>  
</head>  
<body>  
  
  <div id="container">  
    <div id="element"></div>  
  </div>  
<script>  
var ele = document.getElementById('container');  
if(ele) {  
  ele.style.visibility = "visible";  
}  
</script>
```

```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Size for Circular Gauge

You can also set size for the gauge directly through [width](#) and [height](#) properties.

In Pixel

You can set the size of the gauge in pixel as demonstrated below.

INDEX.JS

```
var circulargauge = new ej.circulargauge.CircularGauge({
  // Width and height for gauge in pixel.
  width: '650', height: '350'
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

In Percentage

By setting value in percentage, gauge gets its dimension with respect to its container. For example, when the height is '50%', gauge renders to half of the container height.

INDEX.JS


```
var circulargauge = new ej.circulargauge.CircularGauge({
  // Width and height for gauge in percentage.
  width: '80%', height: '50%'
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  popups/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Note: When you do not specify the size, it takes 450px as the height and window size as its width.

Gauge axes in ##Platform_Name## Circular gauge control

By default, gauge will be displayed with an axis. Each axis contains its own ranges, pointers and annotation.

Axis Customization

You can customize the width and color of an axis line by using [lineStyle](#) property.

Background for an axis can be customized by using [background](#) property.

INDEX.JS

```
var circulargauge = new ej.circulargauge.CircularGauge({
  axes: [{
    lineStyle: {
      width: 2,
```

```

        color: 'red'
    },
    background: 'rgba(0, 128, 128, 0.3)'
  }]
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  popups/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Angles and Direction

Circular gauge axis can sweep from 0 to 360 degrees. By default start angle of an axis is 200 degree and end angle is 160 degree and you can customize this option by using [startAngle](#) and [endAngle](#) property.

INDEX.JS

```

var circulargauge = new ej.circulargauge.CircularGauge({
  axes: [{
    startAngle: 270,
    endAngle: 90
  }]
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

The [direction](#) property enables you to render the gauge axis either in **ClockWise** or in **AntiClockWise** direction.

INDEX.JS

```

var circulargauge = new ej.circulargauge.CircularGauge({
  axes: [{
    direction: 'AntiClockWise'
  }]
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>

```

```
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Axis Radius

By default, radius of an axis is calculated based on the available size.

You can customize this, by using [radius](#) property.

It takes value either in **percentage** or in **pixel**.

In Pixel

You can set the radius of the gauge in pixel as demonstrated below,

INDEX.JS

```
var circulargauge = new ej.circulargauge.CircularGauge({
    axes: [{
        radius: '150'
    }]
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
```

```

        <div id="element"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

In Percentage

By setting value in percentage, gauge gets its dimension with respect to its available size.

For example, when the radius is '50%', gauge renders to half of the available size.

INDEX.JS

```

var circulargauge = new ej.circulargauge.CircularGauge({
    axes: [{
        radius: '50%'
    }]
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Ticks

You can customize the [height](#),

[color](#) and [width](#) of major ticks and minor ticks by using [majorTicks](#) and [minorTicks](#) property.

By default, [interval](#) for

[majorTicks](#) will be calculated automatically and also you can customize the interval for major and minor ticks using [interval](#) property.

INDEX.JS

```
var circulargauge = new ej.circulargauge.CircularGauge({
  axes: [{
    majorTicks: {
      interval: 10,
      color: 'red',
      height: 10,
      width: 3
    },
    minorTicks: {
      interval: 5,
      color: 'green',
      height: 5,
      width: 2
    }
  }]
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  popups/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
```

```

    ele.style.visibility = "visible";
  }
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Tick Position

Both minor and major ticks can be moved by using [offset](#) and [position](#) property. The [offset](#) defines the distance between the axis and ticks.

By default, offset value is 0. The [position](#) will place the ticks either inside or outside of the axis.

By default, ticks will be placed **inside** the axis.

INDEX.JS

```

var circulargauge = new ej.circulargauge.CircularGauge({
  axes: [{
    majorTicks: {
      interval: 10,
      color: 'red',
      height: 10,
      width: 3,
      position: 'Inside',
      offset: 5
    },
    minorTicks: {
      interval: 5,
      color: 'green',
      height: 5,
      width: 2,
      position: 'Inside',
      offset: 5
    }
  }
  ],
  element: '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  popups/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>

```

```
<body>

  <div id="container">
    <div id="element"></div>
  </div>
</body>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Labels

Labels of an axis can be customized by using [font](#) property in [labelStyle](#) options.

INDEX.JS

```
var circulargauge = new ej.circulargauge.CircularGauge({
  axes: [{
    labelStyle: {
      font: {
        color: 'red',
        size: '20px',
        fontWeight: 'Bold'
      }
    }
  }]
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  popups/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
</body>
<script>
var ele = document.getElementById('container');
```



```

if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Label Position

Labels can be moved by using [offset](#) or [position](#) property.

The [offset](#) defines the distance between the labels and ticks.

By default, offset value is 0.

The [position](#) will place the labels either inside or outside of the axis.

By default, labels will be placed **inside** the axis.

INDEX.JS

```

var circulargauge = new ej.circulargauge.CircularGauge({
    axes: [{
        labelStyle: {
            position: 'Outside',
            offset: 5
        }
    }]
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";

```

```

}
    </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Display the last label, even if it isn't in the visible range

If the last label is not in the visible range, it will be hidden by default. If you want to show the last label, set the `showLastLabel` property to `true` in the `axes` API of circular gauge.

INDEX.JS

```

var circulargauge = new ej.circulargauge.CircularGauge({
  axes: [{
    showLastLabel: true,
    minimum: 0,
    maximum: 170,
    startAngle: 210, endAngle: 150
  }]
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  popups/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Auto Angle

Labels can be swept along the axis angle by enabling [autoAngle](#) property.

INDEX.JS

```
var circulargauge = new ej.circulargauge.CircularGauge({
  axes: [{
    labelStyle: {
      autoAngle: true
    }
  }]
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  popups/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Smart Labels

When an axis makes a complete circle, then the first and last label of the axis will get overlap with each other.

In this scenario, you can either hide 1st or last label using [hiddenLabel](#) property.

When `hiddenLabel` value is `First`, then the 1st label will be hidden and when the `hiddenLabel` value is `Last`, then the last label will be hidden.

INDEX.JS

```
var circulargauge = new ej.circulargauge.CircularGauge({
  axes: [{
    minimum: 0,
    maximum: 12,
    startAngle: 0,
    endAngle: 360,
    majorTicks: {
      interval: 1,
      position: 'Inside',
      height: 10
    },
    minorTicks: {
      interval: 0.2,
      position: 'Inside',
      height: 5
    },
    labelStyle: {
      position: 'Inside',
      hiddenLabel: 'First'
    }
  }]
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  popups/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Label Format

Axis labels can be formatted by using [format](#) property in [labelStyle](#) and its supports all globalize format.

INDEX.JS

```
var circulargauge = new ej.circulargauge.CircularGauge({
  axes: [{
    labelStyle: {
      format: 'p1'
    }
  }]
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  popups/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

The following table describes the result of applying some commonly used label formats on numeric values.

Label Value	Label Format property value	Result	Description
1000	n1	1000.0	The Number is rounded to 1 decimal place
1000	n2	1000.00	The Number is rounded to 2 decimal place
1000	n3	1000.000	The Number is rounded to 3 decimal place

0.01	p1	1.0%	The Number is converted to percentage with 1 decimal place
0.01	p2	1.00%	The Number is converted to percentage with 2 decimal place
0.01	p3	1.000%	The Number is converted to percentage with 3 decimal place
1000	c1	\$1,000.0	The Currency symbol is appended to number and number is rounded to 1 decimal place
1000	c2	\$1,000.00	The Currency symbol is appended to number and number is rounded to 2 decimal place

Custom Label Format

Axis labels support custom label format using placeholder like {value}°C, in which the value represent the axis label e.g 20°C.

INDEX.JS

```
var circulargauge = new ej.circulargauge.CircularGauge({
  axes: [{
    labelStyle: {
      format: '{value}°C'
    }
  }]
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  popups/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
</script>
```

```
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Hide intersecting axis labels

When the axis labels overlap with each other, you can hide the intersected labels by setting the `hideIntersectingLabel` property to true in the axis.

INDEX.JS

```
var circulargauge = new ej.circulargauge.CircularGauge({
    axes: [{
        hideIntersectingLabel: true,
        minimum: 0,
        maximum: 200,
        startAngle: 270, endAngle: 90,
        majorTicks:{
            interval:4
        },
        minorTicks: {
            interval:2
        }
    }]
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
```

```

    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Minimum and Maximum

The [minimum](#) and [maximum](#) properties

enables you to customize the start and end values of an axis.

INDEX.JS

```

var circulargauge = new ej.circulargauge.CircularGauge({
  axes: [{
    minimum: 50,
    maximum: 250
  }]
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  popups/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Multiple Axes

In addition to the default axis, you can add n number of axis to a gauge.

Each axis will have its own ranges, pointers, annotations and customization options.

INDEX.JS

```
var circulargauge = new ej.circulargauge.CircularGauge({
  axes: [{
    majorTicks: {
      interval: 10,
      position: 'Inside',
      height: 10,
    },
    pointers: [],
    minorTicks: {
      interval: 5,
      position: 'Inside',
      height: 5,
    }
  }, {
    pointers: [],
    majorTicks: {
      interval: 10,
      position: 'Inside',
      height: 10,
      color: '#27d5ff'
    },
    minorTicks: {
      interval: 5,
      position: 'Inside',
      height: 5,
      color: '#27d5ff'
    }
  }
], '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  popups/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
```

```
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Gauge ranges in ##Platform_Name## Circular gauge control

You can categories certain interval on gauge axis using [ranges](#) property.

Start and End

Start and end value of a range in an axis can be customized by using [start](#) and [end](#) properties.

INDEX.JS

```
var circulargauge = new ej.circulargauge.CircularGauge({
    axes: [{
        ranges: [{
            start: 40,
            end: 80
        }]
    }]
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
```

```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Customization

Color and thickness of the range can be customized by using [color](#), [startWidth](#) and [endWidth](#) property.

INDEX.JS

```
var circulargauge = new ej.circulargauge.CircularGauge({
  axes: [{
    ranges: [{
      start: 40,
      end: 80, endWidth: 15,
      startWidth: 15,
      color: '#ff5985'
    }]
  }]
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  popups/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Radius

You can place the range inside or outside of the axis by using [radius](#) property. The radius of the range can takes value either in percentage or in pixels. By default, ranges

take 100% of the axis radius.

In Pixel

You can set the radius of the range in pixel as demonstrated below,

INDEX.JS

```
var circulargauge = new ej.circulargauge.CircularGauge({
  axes: [{
    minimum: 0,
    maximum: 100,
    ranges: [{
      start: 40,
      end: 80,
      radius: '100'
    }]
  }]
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  popups/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

In Percentage

By setting value in percentage, range gets its dimension with respect to its axis radius. For example, when the radius is '50%', range renders to half of the axis radius.

INDEX.JS

```
var circulargauge = new ej.circulargauge.CircularGauge({
  axes: [{
    minimum: 0,
    maximum: 100,
    ranges: [{
      start: 40,
      end: 80,
      radius: '50%'
    }]
  }]
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  popups/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Dragging Range

The ranges can be dragged over the axis line by clicking and dragging the same. To enable or disable the range drag, use the [enableRangeDrag](#) property.

INDEX.JS

```
var circulargauge = new ej.circulargauge.CircularGauge({
  enableRangeDrag: true,
  height: '250px',
  width: '250px',
```

```

    axes: [{
      ranges: [{
        start: 0,
        end: 100,
        startWidth: 8, endWidth: 8,
        radius: '108%',
        color: '#30B32D'
      }],
      pointers: [{
        value: 50
      }]
    }]
  }, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  popups/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Multiple Ranges

You can add multiple ranges to an axis with the above customization as demonstrated below.

Note: You can set the range color to axis ticks and labels by enabling `useRangeColor` property in [majorTicks](#),

[minorTicks](#) and [labelStyle](#) object.

INDEX.JS

```

var circulargauge = new ej.circulargauge.CircularGauge({
  axes: [{
    minimum: 0,
    maximum: 100,
    majorTicks: {
      useRangeColor: true
    },
    minorTicks: {
      useRangeColor: true
    },
    labelStyle: {
      useRangeColor: true
    },
    ranges: [{
      start: 0,
      end: 25,
      radius: '108%'
    }, {
      start: 25,
      end: 50,
      radius: '70%'
    }, {
      start: 50,
      end: 75,
      radius: '70%'
    }, {
      start: 75,
      end: 100,
      radius: '108%'
    }
  ]
}],
  '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  popups/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>

```

```
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Rounded corner radius

You can customize the corner radius using the `roundedCornerRadius` property in `ranges`.

INDEX.JS

```
var circulargauge = new ej.circulargauge.CircularGauge({
  axes: [{
    minimum: 0,
    maximum: 100,
    ranges: [{
      start: 40,
      end: 80,
      radius: '50%',
      roundedCornerRadius: 5,
    }]
  }]
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  popups/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
```



```

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Gradient Color

Gradient support allows to add multiple colors in the ranges and pointers of the circular gauge. The following gradient types are supported in the circular gauge.

- Linear Gradient
- Radial Gradient

Linear Gradient

Using linear gradient, colors will be applied in a linear progression. The start value of the linear gradient will be set using the [startValue](#) property. The end value of the linear gradient will be set using the [endValue](#) property. The color stop values such as color, opacity and offset are set using [colorStop](#) property.

To apply linear gradient to the range, follow the below code sample.

INDEX.JS

```

var rangeLinearGradient = {
  startValue: '0%', endValue: '100%',
  colorStop: [
    { color: '#9E40DC', offset: '0%', opacity: 0.9 },
    { color: '#E63B86', offset: '70%', opacity: 0.9 }
  ]
};
var circulargauge = new ej.circulargauge.CircularGauge({
  title: 'Short Put Distance',
  titleStyle: {
    size: '18px'
  },
  centerY: '57%',
  axes: [{
    annotations: [{
      content: '12 M', radius: '108%', angle: 98, zIndex: '1'
    }, {
      content: '11 M', radius: '80%', angle: 81, zIndex: '1'
    }, {
      content: '10 M', radius: '50%', angle: 69, zIndex: '1'
    }, {
      content: 'Doe', radius: '108%', angle: 190, zIndex: '1'
    }, {
      content: 'Almaida', radius: '80%', angle: 185, zIndex: '1'
    }, {
      content: 'John', radius: '50%', angle: 180, zIndex: '1'
    }
  ],
  lineStyle: {
    width: 0
  },
  radius: '90%',
  labelStyle: {
    font: {

```

```

        size: '0px'
    },
    majorTicks: {
        width: 0
    },
    minorTicks: {
        width: 0
    },
    startAngle: 200, endAngle: 130,
    minimum: 0, maximum: 14,
    ranges: [{
        start: 0, end: 12, radius: '115%',
        startWidth: 25, endWidth: 25,
        linearGradient : rangeLinearGradient
    }, {
        start: 0, end: 11, radius: '85%',
        startWidth: 25, endWidth: 25,
        linearGradient : rangeLinearGradient
    }, {
        start: 0, end: 10, radius: '55%',
        startWidth: 25, endWidth: 25,
        linearGradient : rangeLinearGradient
    }],
    pointers: [{
        type: 'Marker', value: 12, markerShape: 'Image',
        imageUrl:
'https://ej2.syncfusion.com/javascript/demos/src/circular-gauge/images/football.png',
        radius: '108%', markerWidth: 28, markerHeight: 28,
        animation: { duration: 1500 }
    }, {
        type: 'Marker', value: 11, markerShape: 'Image',
        imageUrl:
'https://ej2.syncfusion.com/javascript/demos/src/circular-gauge/images/basketball.png',
        radius: '78%', markerWidth: 28, markerHeight: 28,
        animation: { duration: 1200 }
    }, {
        type: 'Marker', value: 10, markerShape: 'Image',
        imageUrl:
'https://ej2.syncfusion.com/javascript/demos/src/circular-gauge/images/golfball.png',
        radius: '48%', markerWidth: 28, markerHeight: 28,
        animation: { duration: 900 }
    }, {
        type: 'Marker', value: 12, markerShape: 'Image',
        imageUrl:
'https://ej2.syncfusion.com/javascript/demos/src/circular-gauge/images/athletic.png',
        radius: '0%', markerWidth: 90, markerHeight: 90,
        animation: { duration: 0 }
    }, {
        type: 'Marker', value: 0.1, markerShape: 'Image',
        imageUrl: 'https://ej2.syncfusion.com/demos/src/circular-gauge/images/girl1.png',
        radius: '108%', markerWidth: 28, markerHeight: 28,
        animation: { duration: 1500 }
    }, {

```

```

        type: 'Marker', value: 0.1, markerShape: 'Image',
        imageUrl:
'https://ej2.syncfusion.com/javascript/demos/src/circular-gauge/images/man-
1.png',
        radius: '78%', markerWidth: 28, markerHeight: 28,
        animation: { duration: 1500 }
    }, {
        type: 'Marker', value: 0.1, markerShape: 'Image',
        imageUrl:
'https://ej2.syncfusion.com/javascript/demos/src/circular-gauge/images/man-
2.png',
        radius: '48%', markerWidth: 28, markerHeight: 28,
        animation: { duration: 1500 }
    }
  ]
}
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Radial Gradient

Using radial gradient, colors will be applied in circular progression. The inner circle position of the radial gradient will be set using the [innerPosition](#) property. The outer circle position of the radial gradient can be set using the [outerPosition](#) property. The color stop values such as color, opacity and offset are set using [colorStop](#) property.

To apply radial gradient to the range, follow the below code sample.

INDEX.JS

```
var rangeRadialGradient = {
  radius: '50%', innerPosition: { x: '50%', y: '50%' },
  outerPosition: { x: '50%', y: '50%' },
  colorStop: [
    { color: '#9E40DC', offset: '90%', opacity: 0.9 },
    { color: '#E63B86', offset: '160%', opacity: 0.9 }]
};

var circulargauge = new ej.circulargauge.CircularGauge({
  title: 'Short Put Distance',
  titleStyle: {
    size: '18px'
  },
  centerY: '57%',
  axes: [{
    annotations: [{
      content: '12 M', radius: '108%', angle: 98, zIndex: '1'
    }, {
      content: '11 M', radius: '80%', angle: 81, zIndex: '1'
    }, {
      content: '10 M', radius: '50%', angle: 69, zIndex: '1'
    }, {
      content: 'Doe', radius: '108%', angle: 190, zIndex: '1'
    }, {
      content: 'Almaida', radius: '80%', angle: 185, zIndex: '1'
    }, {
      content: 'John', radius: '50%', angle: 180, zIndex: '1'
    }
  ],
  lineStyle: {
    width: 0
  },
  radius: '90%',
  labelStyle: {
    font: {
      size: '0px'
    }
  },
  majorTicks: {
    width: 0
  },
  minorTicks: {
    width: 0
  },
  startAngle: 200, endAngle: 130,
  minimum: 0, maximum: 14,
  ranges: [{
    start: 0, end: 12, radius: '115%',
    startWidth: 25, endWidth: 25,
    radialGradient: rangeRadialGradient
  }, {
    start: 0, end: 11, radius: '85%',
    startWidth: 25, endWidth: 25,
    radialGradient: rangeRadialGradient
  }, {
    start: 0, end: 10, radius: '55%',
    startWidth: 25, endWidth: 25,
```

```

        radialGradient: rangeRadialGradient
    }],
    pointers: [{
        type: 'Marker', value: 12, markerShape: 'Image',
        imageUrl:
'https://ej2.syncfusion.com/javascript/demos/src/circular-
gauge/images/football.png',
        radius: '108%', markerWidth: 28, markerHeight: 28,
        animation: { duration: 1500 }
    }, {
        type: 'Marker', value: 11, markerShape: 'Image',
        imageUrl:
'https://ej2.syncfusion.com/javascript/demos/src/circular-
gauge/images/basketball.png',
        radius: '78%', markerWidth: 28, markerHeight: 28,
        animation: { duration: 1200 }
    }, {
        type: 'Marker', value: 10, markerShape: 'Image',
        imageUrl:
'https://ej2.syncfusion.com/javascript/demos/src/circular-
gauge/images/golfball.png',
        radius: '48%', markerWidth: 28, markerHeight: 28,
        animation: { duration: 900 }
    }, {
        type: 'Marker', value: 12, markerShape: 'Image',
        imageUrl:
'https://ej2.syncfusion.com/javascript/demos/src/circular-
gauge/images/athletic.png',
        radius: '0%', markerWidth: 90, markerHeight: 90,
        animation: { duration: 0 }
    }, {
        type: 'Marker', value: 0.1, markerShape: 'Image',
        imageUrl:
'https://ej2.syncfusion.com/javascript/demos/src/circular-gauge/images/girl-
1.png',
        radius: '108%', markerWidth: 28, markerHeight: 28,
        animation: { duration: 1500 }
    }, {
        type: 'Marker', value: 0.1, markerShape: 'Image',
        imageUrl:
'https://ej2.syncfusion.com/javascript/demos/src/circular-gauge/images/man-
1.png',
        radius: '78%', markerWidth: 28, markerHeight: 28,
        animation: { duration: 1500 }
    }, {
        type: 'Marker', value: 0.1, markerShape: 'Image',
        imageUrl:
'https://ej2.syncfusion.com/javascript/demos/src/circular-gauge/images/man-
2.png',
        radius: '48%', markerWidth: 28, markerHeight: 28,
        animation: { duration: 1500 }
    }
    ]
    },
    '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

See also

- [Tooltip for Ranges](#)

Gauge pointers in ##Platform_Name## Circular gauge control

Pointers are used to indicate values on the axis. Value of the pointer can be modified using the [value](#) property.

INDEX.JS

```

var circulargauge = new ej.circulargauge.CircularGauge({
  axes: [{
    pointers: [{
      value: 90
    }]
  }]
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">

```

```

<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Gauge supports 3 types of pointers such as **Needle**, **RangeBar** and **Marker**. You can choose any one of the pointer by using **type** property.

Needle Pointers

A needle pointer contains three parts, a needle, a cap / knob and a tail. The length of the needle can be customized by using **radius** property. The length of the tail can be

customized by using **length** property. The radius of the cap can be customized by using **radius** in cap object. The needle and tail

length takes value either in **percentage** or **pixel**.

INDEX.JS

```

var circulargauge = new ej.circulargauge.CircularGauge({
    axes: [{
        pointers: [{
            value: 90,
            radius: '50%',
            cap: {
                radius: 10
            },
            needleTail: {
                length: '25%'
            }
        }
    ]
}]
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Customization

Needle color and width can be customized by using [color](#) and [pointerWidth](#) property.

Cap and tails can be customized by using [cap](#) and [needleTail](#) object.

INDEX.JS

```

var circulargauge = new ej.circulargauge.CircularGauge({
  axes: [{
    pointers: [{
      value: 90,
      radius: '50%',
      cap: {
        radius: 15,
        color: 'white',
        border: {
          color: '#007DD1',
          width: 5
        }
      },
      needleTail: {
        length: '22%',
        color: '#007DD1'
      }
    },
  ],

```



```

        color: '#007DD1',
        pointerWidth: 25
    }
}
}, '#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

The appearance of the needle pointer can be customized by using [needleStartWidth](#) and [needleEndWidth](#).

INDEX.JS

```

var circulargauge = new ej.circulargauge.CircularGauge({
  axes: [{
    pointers: [{
      value: 90,
      radius: '50%',
      cap: {
        radius: 15,
        color: 'white',
        border: {
          color: '#007DD1',
          width: 5
        }
      }
    }
  ]
},
```

```

        needleTail: {
            length: '22%',
            color: '#007DD1'
        },
        color: '#007DD1',
        pointerWidth: 25
    }
}
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  popups/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

RangeBar Pointer

RangeBar pointer is like ranges in an axis, that can be placed on gauge to mark the pointer value.

RangeBar starts from the beginning of the gauge and ends at the pointer value.

INDEX.JS

```

var circulargauge = new ej.circulargauge.CircularGauge({
  axes: [{
    pointers: [{
      value: 50,
      type: 'RangeBar',
      radius: '60%'
    }]
  }]
});

```

```
    }}
  }, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Customization

RangeBar can be customized in terms of color, border and thickness by using

[color](#), [border](#) and [pointerWidth](#) property.

INDEX.JS

```
var circulargauge = new ej.circulargauge.CircularGauge({
  axes: [{
    pointers: [{
      value: 50,
      type: 'RangeBar',
      radius: '60%',
      color: '#007D1',
      border: {
        color: 'grey',
        width: 2
      },
      pointerWidth: 15
    }],
  }],
});
```

```
    }}
  }, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Rounded corner for range bar pointer

The start and end pointers of range bar in the circular gauge are rounded to form arc gauges.

INDEX.JS

```
var circulargauge = new ej.circulargauge.CircularGauge({
  axes: [{
    lineStyle: {
      color: 'transparent'
    },
    ranges: [
      { start: 0, end: 50, color: '#30B32D', radius:'108%' },
      { start: 50, end: 100, color: '#FFDD00', radius:'108%' }
    ],
    pointers: [{
      value: 50,
      type: 'RangeBar',
      roundedCornerRadius: 6
    }]
  }]
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Marker Pointer

Different type of marker shape can be used to mark the pointer value in axis. You can change the marker shape using [markerShape](#) property in pointer. Gauge supports the below marker shape.

- Circle
- Rectangle
- Triangle
- InvertedTriangle
- Diamond

We can use image instead of rendering marker shape to denote the pointer value. It can be achieved by setting [markerShape](#) to Image and assigning image path to [imageUrl](#) in pointer.

INDEX.JS

```
var circulargauge = new ej.circulargauge.CircularGauge({
  axes: [{
    pointers: [{
      value: 90,
      type: 'Marker',
      markerShape: 'InvertedTriangle',
```

```

        radius: '100%',
        markerHeight: 15,
        markerWidth: 15
    }
  }
}, '#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  popups/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
  var ele = document.getElementById('container');
  if(ele) {
    ele.style.visibility = "visible";
  }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Customization

The marker can be customized in terms of color, border, width and height by using

[color](#),

[border](#),

[markerWidth](#) and

[markerHeight](#) property in

[pointer](#).

INDEX.JS

```

var circulargauge = new ej.circulargauge.CircularGauge({
  axes: [{
    pointers: [{
```

```

        value: 90,
        type: 'Marker',
        markerShape: 'Triangle',
        radius: '100%',
        color: 'white',
        border: {
            color: '#007DD1',
            width: 2
        },
        markerHeight: 15,
        markerWidth: 15
    }
}
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  popups/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Dragging pointer

The pointers can be dragged over the axis line by clicking and dragging the same. To enable or disable the pointer drag, use the [enablePointerDrag](#) property.

INDEX.JS

```

var circulargauge = new ej.circulargauge.CircularGauge({
  axes: [{

```

```

        pointers: [{
            value: 90,
            type: 'Marker',
            markerShape: 'Triangle',
            radius: '100%',
            color: 'white',
            border: {
                color: '#007DD1',
                width: 2
            },
            markerHeight: 15,
            markerWidth: 15
        }]
    }
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  popups/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Multiple Pointers

In addition to the default pointer, you can add n number of pointer to an axis by using **pointers** property.

INDEX.JS

```

var circulargauge = new ej.circulargauge.CircularGauge({

```



```

    axes: [{
      pointers: [{
        value: 90,
        type: 'Marker',
        markerShape: 'InvertedTriangle',
        radius: '100%',
        markerHeight: 15,
        markerWidth: 15
      }, {
        value: 90,
        type: 'RangeBar',
        radius: '60%',
        pointerWidth: 10
      }, {
        value: 90,
        radius: '60%',
        cap: {
          radius: 15,
          border: {
            width: 5
          }
        },
        needleTail: {
          length: '22%',
        },
        pointerWidth: 25
      }
    ]
  }],
  '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {

```

```

    ele.style.visibility = "visible";
  }
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Animation

Pointer will get animate on loading the gauge, this can be handled by using

[animation](#) property in pointer.

The [enable](#) property in animation allows you to enable or disable the animation.

The [duration](#) property specify the duration of the animation in milliseconds.

INDEX.JS

```

var circulargauge = new ej.circulargauge.CircularGauge({
  axes: [{
    pointers: [{
      value: 90,
      animation: {
        enable: true,
        duration: 1500
      }
    }]
  }]
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  popups/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";

```

```

}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Gradient Color

Gradient support allows to add multiple colors in the ranges and pointers of the circular gauge. The following gradient types are supported in the circular gauge.

- Linear Gradient
- Radial Gradient

Linear Gradient

Using linear gradient, colors will be applied in a linear progression. The start value of the linear gradient can be set using the [startValue](#) property. The end value of the linear gradient will be set using the [endValue](#) property. The color stop values such as color, opacity and offset are set using [colorStop](#) property.

The linear gradient can be applied to all pointer types like marker, range bar and needle. To do so, follow the below code sample.

INDEX.JS

```

var pointerLinearGradient = {
  startValue: '0%',
  endValue: '100%',
  colorStop: [
    { color: '#FEF3F9', offset: '0%', opacity: 0.9 },
    { color: '#E63B86', offset: '70%', opacity: 0.9 }
  ]
};
var circulargauge = new ej.circulargauge.CircularGauge({
  axes: [{
    startAngle: 270,
    endAngle: 90,
    lineStyle: { width: 3, color: '#E63B86' },
    labelStyle: {
      font: { size: '0px' }
    },
    majorTicks: {
      height: 0,
    },
    minorTicks: {
      height: 0,
    },
  },
  radius: '90%',
  minimum: 0,
  maximum: 100,
  pointers: [{
    radius: '80%',
    value: 80,
    animation: { enable: true, duration: 1000 },
    color: '#e3a21a',
    pointerWidth: 10,
    linearGradient: pointerLinearGradient,
    cap: {

```

```

        radius: 8,
        color: 'white',
        border: {
            color: '#e3a21a',
            width: 1
        }
    },
    needleTail: {
        length: '20%',
        linearGradient: pointerLinearGradient
    }
}, {
    radius: '60%', value: 40,
    animation: { duration: 1000 },
    color: '#ffbb13',
    pointerWidth: 10,
    linearGradient: pointerLinearGradient,
    cap: {
        radius: 8, color: 'white',
        border: { color: '#ffbb13', width: 1 }
    },
    needleTail: {
        length: '20%',
        linearGradient: pointerLinearGradient
    }
}
]]
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  popups/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";

```

```

}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Radial Gradient

Using radial gradient, colors will be applied in circular progression. The inner circle position of the radial gradient will be set using the [innerPosition](#) property. The outer circle position of the radial gradient can be set using the [outerPosition](#) property. The color stop values such as color, opacity and offset are set using [colorStop](#) property.

The radial gradient can be applied to all pointer types like marker, range bar and needle. To do so, follow the below code sample.

INDEX.JS

```

var pointerRadialGradient = {
  radius: '50%',
  innerPosition: { x: '50%', y: '50%' },
  outerPosition: { x: '50%', y: '50%' },
  colorStop: [
    { color: '#FEF3F9', offset: '0%', opacity: 0.9 },
    { color: '#E63B86', offset: '60%', opacity: 0.9 }
  ]
};
var circulargauge = new ej.circulargauge.CircularGauge({
  axes: [{
    startAngle: 270,
    endAngle: 90,
    lineStyle: { width: 3, color: '#E63B86' },
    labelStyle: {
      font: { size: '0px' }
    },
    majorTicks: {
      height: 0
    },
    minorTicks: {
      height: 0
    },
  },
  radius: '90%',
  minimum: 0,
  maximum: 100,
  pointers: [{
    radius: '80%',
    value: 80,
    animation: { enable: true, duration: 1000 },
    pointerWidth: 10,
    radialGradient: pointerRadialGradient,
    cap: {
      radius: 8,
      color: 'white',
      border: {
        color: '#e3a21a',
        width: 1
      }
    },
  },
  needleTail: {
    length: '20%',

```

```

        radialGradient: pointerRadialGradient,
    }, {
        radius: '60%', value: 40,
        animation: { duration: 1000 },
        pointerWidth: 10,
        radialGradient: pointerRadialGradient,
        cap: {
            radius: 8, color: 'white',
            border: { color: '#ffb133', width: 1 }
        },
        needleTail: {
            length: '20%',
            radialGradient: pointerRadialGradient
        }
    }
}
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  popups/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Gauge annotations in ##Platform_Name## Circular gauge control

Annotations are used to mark a specific area of interest in the gauge with texts, shapes or images.

Content

You can place any custom element on the axis area by assigning the id of the element to [content](#) property of [annotation](#) object.

Note: To use annotation feature, we need to inject **Annotations** module using **CircularGauge.Inject(Annotations)** method.

INDEX.JS

```
var circulargauge = new ej.circulargauge.CircularGauge({
  axes: [{
    annotations: [{
      content: 'annotation-template',
      zIndex: '1'
    }],
    pointers: [{
      value: 50
    }]
  }]
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  popups/styles/material.css" rel="stylesheet">

  <script id="annotation-template" type="text/x-template">
    <div id='templateWrap'>
      <div class='des'>
        <span>Pointer Value : 50</span>
      </div>
    </div>
  </script>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
  var ele = document.getElementById('container');
  if(ele) {
    ele.style.visibility = "visible";
  }
  </script>
```

```

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Position

Annotation can be placed around the axis by using [radius](#) and [angle](#) property.

For example, if the angle is 90 degree and the radius is 110%, then the annotation, will be placed at the right side of the axis.

Radius of the annotation takes value either in pixel or percentage. By setting value in percentage, annotation gets its position with respect to its axis radius.

INDEX.JS

```

var circulargauge = new ej.circulargauge.CircularGauge({
  axes: [{
    annotations: [{
      content: '#annotation-template',
      angle: 90,
      radius: '150%',
      zIndex: '1'
    }],
    pointers: [{
      value: 50
    }]
  }]
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  popups/styles/material.css" rel="stylesheet">

  <script id="annotation-template" type="text/x-template">
    <div id='templateWrap'>
      <div class='des'>
        <span>Pointer Value : 50</span>
      </div>
    </div>
  </script>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

```



```

<div id="container">
  <div id="element"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Sub Gauge

As the annotation allows you to place any custom element, we can initialize a gauge to the element and can be used to place that in another gauge.

INDEX.JS

```

var circulargauge = new ej.circulargauge.CircularGauge({
  axes: [{
    minimum: 0,
    maximum: 12,
    startAngle: 0,
    endAngle: 360,
    lineStyle: { width: 0 },
    ranges: [
      {
        start: 0, end: 3,
        color: 'rgba(29,29,29,0.7)'
      }, {
        start: 3, end: 12,
        color: 'rgba(168,145,102,0.1)'
      }
    ],
    annotations: [{
      angle: 270,
      radius: '40%',
      content: '<div id="subGauge"
style="width:90px;height:90px;"></div>'
    }, {
      angle: 90,
      radius: '40%',
      content: '<div id="time"><span>6:30 PM</span></div>'
    }],
    labelStyle: {
      hiddenLabel: 'First'
    },
    pointers: [{
      pointerWidth: 5,
      radius: '40%',
      value: 6.5,
      color: 'rgb(29,29,29)',
      border: { width: 1, color: 'rgb(29,29,29)' },
      cap: {
        color: 'rgb(29,29,29)',
        radius: 0,

```

```

        border: {
            width: 0.2,
            color: 'red'
        }
    },
    needleTail: {
        length: '0%'
    }, animation: {
        enable: false
    }
}, {
    radius: '60%',
    pointerWidth: 5,
    color: 'rgb(29,29,29) ',
    border: {
        width: 1,
        color: 'rgb(29,29,29) '
    },
    value: 6,
    cap: {
        color: 'rgb(29,29,29) ',
        radius: 0,
        border: {
            width: 0.2,
            color: 'red'
        }
    },
    needleTail: {
        length: '0%'
    }, animation: {
        enable: false
    }
}, {
    radius: '70%',
    pointerWidth: 4,
    value: 9.8,
    color: 'rgba(168,145,102,1) ',
    cap: {
        color: 'rgba(168,145,102,1) ',
        radius: 4,
        border: {
            width: 0.2,
            color: 'rgba(168,145,102,1) '
        }
    },
    needleTail: {
        color: 'rgba(168,145,102,1) ',
        length: '20%'
    }, animation: {
        enable: false,
        duration: 500
    }
}
    ]]
    }, '#element');
    var circulargauge = new ej.circulargauge.CircularGauge({
        axes: [{

```

```

    minimum: 0,
    maximum: 12,
    startAngle: 0,
    endAngle: 360,
    majorTicks: {
        interval: 3
    },
    lineStyle: { width: 0 },
    ranges: [
        {
            start: 0, end: 3,
            startWidth: 5, endWidth: 5,
            color: 'rgba(29,29,29,0.7)'
        }, {
            start: 3, end: 12,
            startWidth: 5, endWidth: 5,
            color: 'rgba(168,145,102,0.1)'
        }
    ],
    labelStyle: {
        hiddenLabel: 'First',
        offset: -5
    },
    pointers: [{
        pointerWidth: 2,
        radius: '40%',
        color: 'rgb(29,29,29)',
        border: { width: 1, color: 'rgb(29,29,29)' },
        cap: {
            color: 'rgb(29,29,29)',
            radius: 2,
            border: {
                width: 0.2,
                color: 'red'
            }
        }
    },
    needleTail: {
        length: '0%'
    }, animation: {
        enable: false
    }
  ]
}]
}, '#subGauge');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  popups/styles/material.css" rel="stylesheet">

```

```

<script id="annotation-template" type="text/x-template">
  <div id='templateWrap'>
    <div class='des'>
      <span>Pointer Value : 50</span>
    </div>
  </div>
</script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See also

- [Tooltip for Annotation](#)

Animation in ##Platform_Name## Circular Gauge control

All of the elements in the Circular Gauge, such as the axis lines, ticks, labels, ranges, pointers, and annotations, can be animated sequentially by using the [animationDuration](#) property. The animation for the Circular Gauge is enabled when the `animationDuration` property is set to an appropriate value in milliseconds, providing a smooth rendering effect for the control. If the `animationDuration` property is set to `0`, which is the default value, the animation effect is disabled. If the animation is enabled, the control will behave in the following order.

1. The axis line will be animated in the rendering direction (clockwise or anticlockwise).
2. Each tick line and label will then be animated.
3. If available, ranges will be animated.
4. If available, annotations will be animated.

The animation of the Circular Gauge is demonstrated in the following example.

INDEX.JS

```

var circulargauge = new ej.circulargauge.CircularGauge({
  animationDuration: 2000,
  axes: [

```

```

{
  annotations: [
    {
      angle: 165,
      radius: '35%',
      zIndex: 1,
      content:
        '<div style="font-size:18px;margin-left: -
20px;margin-top: -12px; color:#9DD55A">60</div>',
    },
    ],
    radius: '80%',
    startAngle: 230,
    endAngle: 130,
    majorTicks: {
      offset: 5,
    },
    lineStyle: { width: 8, color: '#E0E0E0' },
    minorTicks: {
      offset: 5,
    },
    labelStyle: {
      font: {
        fontFamily: 'inherit',
      },
      offset: -1,
    },
    pointers: [
      {
        value: 60,
        radius: '60%',
        pointerWidth: 7,
        cap: {
          radius: 8,
          color: '#c06c84',
          border: { width: 0 },
        },
        needleTail: {
          length: '0%',
        },
        color: '#c06c84',
        animation: {
          enable: true,
          duration: 500,
        },
      },
    ],
    ranges: [
      {
        start: 0,
        end: 30,
        color: '#E63B86',
        startWidth: 22,
        endWidth: 22,
        radius: '60%',
        linearGradient: {
          startValue: '0%',

```

```

        endValue: '100%',
        colorStop: [
            { color: '#9e40dc', offset: '0%', opacity: 1 },
            { color: '#d93c95', offset: '70%', opacity: 1 },
        ],
    },
},
{
    start: 30,
    end: 60,
    color: '#E0E0E0',
    startWidth: 22,
    endWidth: 22,
    radius: '60%',
},
],
},
});
circulargauge.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  popups/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Gauge legend in ##Platform_Name## Circular gauge control

Legend provides valuable information for interpreting what the circular gauge axis range displays, and they can be represented in various colors, shapes, and other identifiers based on the data. It gives a breakdown of what each symbol represents in the axis range of circular gauge.

You can add the legend for circular gauge ranges by setting the visible property of `legendSettings` to true.

Legend customization

Customization option is also provided for the legend shape, alignment, and position.

Position and alignment

The position of the legend is used to place legend in various positions. You can use the `position` property in `legendSettings`. Based on the position, the legend item will be aligned. The following options are available to customize the legend position:

- Top
- Bottom
- Left
- Right
- Custom
- Auto

The legend alignment is used to align the legend items in specific location. You can use the alignment property in `legendSettings` to align the legend items. The following options are available to customize the legend alignment:

- Near
- Center
- Far

The legends can also be positioned to absolute position using the `location.x` and `location.y` properties available in `legendSettings`.

Legend size

The legend size can be modified using the `height` and `width` properties in `legendSettings`.

Legend opacity

To specify the transparency for legend shape, set the `opacity` property in `legendSettings`.

Legend shape

To change the legend item shape, specify the desired shape in the `shape` property of the legend. By default, the shape of the legend is `circle`.

It also supports the following shapes:

- Circle
- Rectangle
- Diamond
- Triangle

- InvertedTriangle
- Image

You can customize a shape using the `shapeWidth` and `shapeHeight` properties.

Legend padding

You can control the spacing between the legend items using the `padding` option of the legend. The default value of padding is 5.

Legend border

You can customize the legend border using the border option in the legend. The legend border can be customized using the border `color` and `width` properties.

Font of the legend text

The font of the legend item text can be customized using the following properties:

- fontFamily
- fontStyle
- fontWeight
- opacity
- color
- size

The following code example shows how to add legend in the gauge.

INDEX.JS

```
var circulargauge = new ej.circulargauge.CircularGauge({
  axes: [{
    minimum: 0,
    maximum: 100,
    majorTicks: {
      useRangeColor: true
    },
    minorTicks: {
      useRangeColor: true
    },
    labelStyle: {
      useRangeColor: true
    },
    ranges: [{
      start: 0,
      end: 25,
      radius: '108%'
    }, {
      start: 25,
      end: 50,
      radius: '108%'
    }, {
      start: 50,
      end: 75,
      radius: '108%'
    }, {
      start: 75,
```



```

        end: 100,
        radius: '108%'
    }
  ]],
  legendSettings : {
    visible: true,
    position: 'Bottom',
    alignment: 'Near',
    width: '450',
    height: '80',
    shapeWidth: 30,
    shapeHeight: 30,
    padding: 15,
    border: {
      color: 'green',
      width: 3
    }
  }
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Toggle option in legend

The toggle option has been provided for legend. So, if you toggle the legend, the given color will be changed to the corresponding circular gauge range. You can enable the toggle option using [toggleVisibility](#) in the `legendSettings` property.

INDEX.JS

```
var circulargauge = new ej.circulargauge.CircularGauge({
  axes: [{
    minimum: 0,
    maximum: 100,
    majorTicks: {
      useRangeColor: true
    },
    minorTicks: {
      useRangeColor: true
    },
    labelStyle: {
      useRangeColor: true
    },
    ranges: [{
      start: 0,
      end: 25,
      radius: '108%'
    }, {
      start: 25,
      end: 50,
      radius: '108%'
    }, {
      start: 50,
      end: 75,
      radius: '108%'
    }, {
      start: 75,
      end: 100,
      radius: '108%'
    }
  ]
}],
  legendSettings : {
    visible: true,
    toggleVisibility: true
  }
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  popups/styles/material.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Paging support in legend

By default, paging will be enabled if the legend items exceed the legend bounds. You can view each legend item by navigating between the pages using navigation buttons.

INDEX.JS

```

var circulargauge = new ej.circulargauge.CircularGauge({
axes: [{
    minimum: 0,
    maximum: 100,
    majorTicks: {
        useRangeColor: true
    },
    minorTicks: {
        useRangeColor: true
    },
    labelStyle: {
        useRangeColor: true
    },
    ranges: [{
        start: 0,
        end: 25,
        radius: '108%'
    }, {
        start: 25,
        end: 50,
        radius: '108%'
    }, {
        start: 50,
        end: 75,
        radius: '108%'
    }, {
        start: 75,
        end: 100,
        radius: '108%'
    }
}]

```

```

    }],
    legendSettings : {
      visible: true,
      height: '50'
    }
  }, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  popups/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Legend text customization

You can customize the legend text using [legendText](#) property in `ranges`.

INDEX.JS

```

var circulargauge = new ej.circulargauge.CircularGauge({
  axes: [{
    minimum: 0,
    maximum: 100,
    majorTicks: {
      useRangeColor: true
    },
    minorTicks: {
      useRangeColor: true
    },
    labelStyle: {

```

```

        useRangeColor: true
    },
    ranges: [{
        start: 0,
        end: 25,
        radius: '108%',
        legendText: 'light air'
    }, {
        start: 25,
        end: 50,
        radius: '108%',
        legendText: 'light air'
    }, {
        start: 50,
        end: 75,
        radius: '108%',
        legendText: 'light breeze'
    }, {
        start: 75,
        end: 100,
        radius: '108%',
        legendText: 'Gentle breeze'
    }
    ]
},
legendSettings : {
    visible: true
}
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";

```

```

}
    </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

legendRendering event will be triggered before rendering each legend item, using this event you can customize needed legend items using following arguments.

Argument name	Description
fill	Specifies the legend shape color
text	Specifies the current legend text
shape	Customize the shape of the legends
name	Specifies the name of the event
cancel	Set to true, to cancel the event status

Gauge user interaction in ##Platform_Name## Circular gauge control

Tooltip for pointers

Circular gauge will displays the pointer details through [tooltip](#),when the mouse is moved over the pointer.

Enable Tooltip

By default, tooltip is not visible. Enable the tooltip by setting [enable](#) property to true and injecting GaugeTooltip module using `CircularGauge.Inject(GaugeTooltip)` method.

INDEX.JS

```

var circulargauge = new ej.circulargauge.CircularGauge({
  // Title for circular gauge.
  tooltip: {
    enable: true
  },
  axes:[{
    pointers:[{
      value: 70
    }],
  }],
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  popups/styles/material.css" rel="stylesheet">

```

```

<script id="template-tooltip" type="text/x-template">
  <div id='templateWrap'>
    <div class='des' style="float: right; padding-left:10px; line-
height:30px;">
      <span>Pointer &#160;&#160;:&#160;
    </div>
    <div>
      <span>Pointer &#160;&#160;:&#160;
    </div>
  </div>
</script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Template

Any HTML elements can be displayed in the tooltip by using the [template](#) property of the tooltip.

INDEX.JS

```

var circulargauge = new ej.circulargauge.CircularGauge({
  // Title for circular gauge.
  tooltip: {
    enable: true,
    template: '${value}'
  },
  axes:[{
    pointers:[{
      value: 70
    }]
  }]
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">

```

```

<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

<script id="template-tooltip" type="text/x-template">
  <div id='templateWrap'>
    <div class='des' style='float: right; padding-left:10px; line-
height:30px;'>
      <span>Pointer &#160;&#160;:&#160;
      ${Math.round(pointers[0].value)}</span>
    </div>
  </div>
</script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Tooltip for ranges

Circular gauge displays the information about the ranges through tooltip when hovering the mouse over the ranges. You can enable this feature by setting the type property of tooltip to 'Range' in the array collection.

Range tooltip customization

To customize the range tooltip, use the `rangeSettings` property in tooltip. The following options are available to customize the range tooltip:

- `fill` - Specifies the range tooltip fill color.
- `textStyle` - Specifies the range tooltip text style.
- `format` - Specifies the range content format.
- `template` - Specifies the custom template for tooltip.
- `enableAnimation` - Animates as it moves from one point to another.
- `border` - Specifies the tooltip border.
- `showMouseAtPosition` - Displays the position of the tooltip on the cursor position.

Tooltip for annotation

Circular gauge displays the information about the annotations through tooltip when hovering the mouse over the annotation. You can enable this feature by setting the type property of tooltip to 'Annotation' in the array collection.

Annotation tooltip customization

To customize the annotation tooltip, use the `annotationSettings` property in tooltip. The following options are available to customize the annotation tooltip:

- `fill` - Specifies the annotation tooltip fill color.
- `textStyle` - Specifies the annotation tooltip text style.
- `format` - Specifies the annotation content format.
- `template` - Specifies the tooltip content with custom template.
- `enableAnimation` - Animates as it moves from one point to another.
- `border` - Specifies the tooltip border.

The following code example shows the tooltip for the ranges and annotation.

INDEX.JS

```
var circulargauge = new ej.circulargauge.CircularGauge({
  // Title for circular gauge.
  axes: [{
    radius: '90%',
    minimum: 0,
    maximum: 120,
    startAngle: 240,
    endAngle: 120,
    annotations: [{
      content: 'CircularGauge', zIndex: '1', angle: 180
    }],
   LineStyle: { width: 0 },
    majorTicks: { color: 'white', offset: -5, height: 12 },
    minorTicks: { width: 0 },
    labelStyle: { useRangeColor: true, font: { color: '#424242', size:
'13px', fontFamily: 'Roboto' } },
    pointers: [{
      value: 70,
      radius: '60%',
      color: '#33BCBD',
      cap: { radius: 10, border: { color: '#33BCBD', width: 5 } },
      animation: { enable: false }
    }],
    ranges: [{
      start: 0,
      end: 50,
      startWidth: 10, endWidth: 10,
      radius: '102%',
      color: '#3A5DC8',
    }, {
      start: 50,
      end: 120,
      radius: '102%',
      startWidth: 10, endWidth: 10,
      color: '#33BCBD',
```

```

    ]]
  }],
  tooltip: {
    type: ['Pointer', 'Range', 'Annotation'],
    enable: true,
    enableAnimation: false,
    annotationSettings: { template: '<div>CircularGauge</div>' },
    rangeSettings: { fill: 'red' }
  }
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  popups/styles/material.css" rel="stylesheet">

  <script id="template-tooltip" type="text/x-template">
    <div id='templateWrap'>
      <div class='des' style="float: right; padding-left:10px; line-
  height:30px;">
        <span>Pointer &#160;&#160;:&#160;
  ${Math.round(pointers[0].value)}</span>
      </div>
    </div>
  </script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Pointer Drag

Pointers can be dragged over the axis value. This can be achieved by clicking and dragging the pointer. To enable or disable the pointer drag, you can use

[enablePointerDrag](#) property.

INDEX.JS

```
var circulargauge = new ej.circulargauge.CircularGauge({
  enablePointerDrag: true,
  tooltip: {
    enable: true
  },
  axes:[{
    pointers:[{
      value: 70
    }]
  }]
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  popups/styles/material.css" rel="stylesheet">

  <script id="template-tooltip" type="text/x-template">
    <div id='templateWrap'>
      <div class='des' style="float: right; padding-left:10px; line-
      height:30px;">
        <span>Pointer &#160;&#160;:&#160;
        ${Math.round(pointers[0].value)}</span>
      </div>
    </div>
  </script>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Gauge appearance in ##Platform_Name## Circular gauge control

Gauge Title

Circular gauge can be given a title by using [title](#) property, to show the information about the gauge.

Title can be customized by using [titleStyle](#) property in gauge.

INDEX.JS

```
var circulargauge = new ej.circulargauge.CircularGauge({
  title: 'Speedometer',
  titleStyle: {
    color: '#27d5ff'
  }
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Gauge Position

Gauge can be positioned anywhere in the container with the help of [centerX](#) and [centerY](#) property and it accepts values either in percentage or in pixels.

The default value of the [centerX](#) and

[centerY](#) property is 50%, which means gauge will get rendered to the centre of the container.

In Pixel

You can set the mid point of the gauge in pixel as demonstrated below,

INDEX.JS

```
var circulargauge = new ej.circulargauge.CircularGauge({
  centerX: '20',
  centerY: '20',
  axes: [{
    lineStyle: {
      width: 2,
      color: '#F8F8F8'
    },
    startAngle: 90,
    endAngle: 180
  }]
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  popups/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

In Percentage

By setting the value in percentage, gauge gets its mid point with respect to its plot area.

For example, when the [centerX](#) value as '0%' and [centerY](#) value is '50%', gauge will get positioned at the top left corner of the plot area.

INDEX.JS

```
var circulargauge = new ej.circulargauge.CircularGauge({
  centerX: '10%',
  centerY: '50%',
  axes: [{
    lineStyle: {
      width: 2,
      color: '#F8F8F8'
    },
    startAngle: 0,
    endAngle: 180
  }]
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  popups/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Area Customization

Customize the gauge background

Using [background](#) and [border](#) properties, you can change the background color and border of the circular gauge.

INDEX.JS

```
var circulargauge = new ej.circulargauge.CircularGauge({
  background: 'skyblue',
  // Customize the chart border and opacity.
  border: {color: "#FF0000", width: 2},
  axes: [{
    radius: '90%',
    maximum: 120,
    startAngle: 230,
    endAngle: 130,
    majorTicks: {
      width: 1, color: '#8c8c8c'
    },
    lineStyle: { width: 2 },
    minorTicks: {
      width: 1, color: '#8c8c8c'
    },
    pointers: [{
      value: 60,
      radius: '60%'
    }],
    ranges: [{
      start: 0,
      end: 70,
      radius: '110%',
      strokeWidth: 10
    }, {
      start: 70,
      end: 110,
      radius: '110%',
      strokeWidth: 10
    }, {
      start: 110,
      end: 120,
      radius: '110%',
      strokeWidth: 10
    }
  ]
}],
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  popups/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Gauge Margin

You can set margin for gauge from its container through [margin](#) property.

INDEX.JS

```

var circulargauge = new ej.circulargauge.CircularGauge({
    background: 'skyblue',
    // Customize the chart border and opacity.
    border: {color: "#FF0000", width: 2},
    // Change chart margin to left, right, top and bottom.
    margin: { left: 40, right: 40, top: 40, bottom: 40 },
    axes: [{
        radius: '90%',
        maximum: 120,
        startAngle: 230,
        endAngle: 130,
        majorTicks: {
            width: 1, color: '#8c8c8c'
        },
        lineStyle: { width: 2 },
        minorTicks: {
            width: 1, color: '#8c8c8c'
        },
        pointers: [{
            value: 60,
            radius: '60%'
        }],
        ranges: [{
            start: 0,
            end: 70,
            radius: '110%',
            strokeWidth: 10
        }, {
            start: 70,
            end: 110,
            radius: '110%',
            strokeWidth: 10
        }, {
            start: 110,

```



```

        end: 120,
        radius: '110%',
        strokeWidth: 10
    }]
    }]
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  popups/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Radius calculation based on angles

Render semi or quarter circular gauges by modifying the start and end angles. By enabling the radius based on angle option, the radius of circular gauge will be calculated based on the start and end angles to avoid excess white space.

INDEX.JS

```

var circulargauge = new ej.circulargauge.CircularGauge({
  moveToCenter: true,
  axes: [{
    lineStyle: {
      width: 2,
      color: '#F8F8F8'
    },
    startAngle: 270,
    endAngle: 90,
  }

```

```
        radius: '80%'
    }]
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Accessibility in ##Platform_Name## Circular gauge control

Circular Gauge has built-in accessibility features like screen reading and WAI-ARIA attributes.

WAI-ARIA attributes

The Circular Gauge control followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Circular Gauge control:

| Attributes | Purpose |

| --- | --- |

| **role=region** | It is specified in the pointer where the interactive drag and drop function is supported to update the pointer value. |

| **aria-label** | Provides an accessible name for the axis labels, legend title, legend item label, text pointer and annotation. |

Screen reading in Circular Gauge

Accessibility in the Circular Gauge control ensures that all users, regardless of ability or disability, can use screen reading. The following Circular Gauge elements will be read aloud using screen reading software, such as Narrator for Windows.

Elements	Description
---	---
Axis labels	Reads the axis labels of the Circular Gauge.
Legend title	Reads the title of the legend in the Circular Gauge.
Legend item label	Reads the label of the legend item in the Circular Gauge.
Text pointer	Reads the text content shown as a pointer in Circular Gauge.
Annotation	Reads the content specified in the annotation.

Ensuring accessibility

The Circular Gauge control's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Circular Gauge control is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Circular Gauge control with accessibility tools.

See also

- [Accessibility in Syncfusion ##Platform_Name## controls](#)

Internationalization in ##Platform_Name## Circular gauge control

Circular Gauge provides internationalization support for below elements.

- Axis Labels
- Tooltip

For more information about number formatter, you can refer [internationalization](#).

Globalization

Globalization is the process of designing and developing a control that works in different cultures/locales.

Internationalization library is used to globalize number in the Circular Gauge using [format](#) property in [labelStyle](#).

Numeric Format

In the below example, axis labels are globalized to **EUR**.

INDEX.JS

```
var circulargauge = new ej.circulargauge.CircularGauge({
  axes: [{
    labelStyle: {
      position: 'Inside',
      //Label format as currency.
```

```

        format: 'c'
    }
    }]
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  popups/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Right-to-left

Circular Gauge can render its elements from right to left, which improves the user experience for certain language users. To do so, set the [enableRtl](#) property to **true**. When this property is enabled, elements such as the tooltip and legend will be rendered from right to left. Meanwhile, the axis can be rendered from right to left by setting the [direction](#) property to **AntiClockWise**. For more information on axis, click [here](#).

The following example illustrates the right to left rendering of the Circular Gauge.

INDEX.JS

```

var circulargauge = new ej.circulargauge.CircularGauge({
  enableRtl:true,
  tooltip: {
    type:['Pointer', 'Range'],
    format:'Pointer : {value} ',
    enable: true,
    enableAnimation: false

```

```

    },
    legendSettings: {
        visible: true
    },
    axes: [{
        direction: 'AntiClockWise',
        lineStyle: { width: 10, color: 'transparent' },
        labelStyle: {
            position: 'Inside', useRangeColor: false,
            font: {
                size: '12px',
                color: '#424242',
                fontFamily: 'Roboto',
                fontStyle: 'Regular'
            }
        },
    },
    majorTicks: {
        height: 10,
        offset: 5,
        color: '#9E9E9E'
    },
    minorTicks: { height: 0 },
    startAngle: 210,
    endAngle: 150,
    minimum: 0,
    maximum: 120,
    radius: '80%',
    ranges: [{
        start: 0,
        end: 40,
        color: '#30B32D'
    },
    {
        start: 40,
        end: 80,
        color: '#FFDD00'
    },
    {
        start: 80,
        end: 120,
        color: '#F03E3E'
    }
    ],
    pointers: [{
        animation: { enable: false },
        value: 65,
        radius: '60%',
        color: '#757575',
        pointerWidth: 8,
        cap: {
            radius: 7,
            color: '#757575'
        },
        needleTail: {
            length: '18%'
        }
    }
    ]
    ]
}

```

```
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Ej1 api migration in ##Platform_Name## Circular gauge control

This article describes the API migration process of Accordion component from Essential JS 1 to Essential JS 2.

Circular gauge dimensions

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| **Height** | **Property:** *height*

 \$("#container").ejCircularGauge({ height: 400 }); | **Property:** *height*

 let gauge: CircularGauge = new CircularGauge({
 height : '350px'
 });
 gauge.appendTo('#container'); |

| **Width** | **Property:** *width*

 \$("#container").ejCircularGauge({ width: 100 }); | **Property:** *width*

 let gauge: CircularGauge = new CircularGauge({
 width : '150px'
 });
 gauge.appendTo('#container'); |

| **Height(In Percentage)** | Not Applicable | **Property:** *height*

 let gauge: CircularGauge = new CircularGauge({
 height : '50%'
 });
 gauge.appendTo('#container'); |

|Width(In Percentage)| Not Applicable | **Property:** *width*
 let gauge: CircularGauge = new CircularGauge({
 width : '80%'
 });
 gauge.appendTo('#container');

Axis Line

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

|Axisline Width| **Property:** *scales.size*
 \$("#container").ejCircularGauge({
 scales: [{
 showScaleBar: true, size: 6
 }]}); | **Property:** *axes.lineStyle.width*
 let gauge: CircularGauge = new CircularGauge({
 axes: [{
 lineStyle: { width: 4 }
 }]});
 gauge.appendTo('#container');

|Axisline Color| **Property:** *scales.size*
 \$("#container").ejCircularGauge({
 scales: [{
 showScaleBar: true, backgroundColor: "red"
 }]}); | **Property:** *axes.lineStyle.width*
 let gauge: CircularGauge = new CircularGauge({
 axes: [{
 lineStyle: { color: 'red' }
 }]});
 gauge.appendTo('#container');

|Axisline BackgroundColor| Not Applicable | **Property:** *axes.background*
 let gauge: CircularGauge = new CircularGauge({
 axes: [{ background: 'red' }]
 });
 gauge.appendTo('#container');

|Axisline Direction| **Property:** *scales.direction*
 \$("#container").ejCircularGauge({
 scales: [{
 direction: "counterclockwise"
 }]}); | **Property:** *axes.direction*
 let gauge: CircularGauge = new CircularGauge({
 axes: [{ direction: 'AntiClockWise' }]
 });
 gauge.appendTo('#container');

|Axisline Radius| **Property:** *scales.radius*
 \$("#container").ejCircularGauge({
 scales: [{
 showScaleBar: true, radius: 150
 }]}); | **Property:** *axes.radius*
 let gauge: CircularGauge = new CircularGauge({
 axes: [{ radius: '150' }]
 });
 gauge.appendTo('#container');

|Axisline Startangle| **Property:** *scales.startAngle*
 \$("#container").ejCircularGauge({
 scales: [{
 startAngle: 80
 }]}); | **Property:** *axes.startAngle*
 let gauge: CircularGauge = new CircularGauge({
 axes: [{ startAngle: 200 }]
 });
 gauge.appendTo('#container');

|Axisline Endangle| **Property:** *scales.sweepAngle*
 \$("#container").ejCircularGauge({
 scales: [{
 sweepAngle: 250
 }]}); | **Property:** *axes.endAngle*
 let gauge: CircularGauge = new CircularGauge({
 axes: [{ endAngle: 150 }]
 });
 gauge.appendTo('#container');

|Minimum Axisvalue| **Property:** *scales.minimum*
 \$("#container").ejCircularGauge({
 scales: [{
 minimum: 20
 }]}); | **Property:** *axes.minimum*
 let gauge: CircularGauge = new CircularGauge({
 axes: [{ minimum: 20 }]
 });
 gauge.appendTo('#container');

|Maximum Axisvalue| **Property:** *scales.maximum*
 \$("#container").ejCircularGauge({
 scales: [{
 maximum: 200
 }]}); | **Property:** *axes.maximum*
 let gauge: CircularGauge = new CircularGauge({
 axes: [{ maximum: 200 }]
 });
 gauge.appendTo('#container');

Ticks

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

|Type of Ticks| **Property:** *scales.ticks.type*
 \$("#container").ejCircularGauge({
 scales: [{ type: "major" }];
 | **Property:** *axes.majorTicks*
 let gauge: CircularGauge = new CircularGauge({
 axes: {
 majorTicks: { }
 };
 gauge.appendTo('#container');

|Height of Major Ticks| **Property:** *scales.ticks.height*
 \$("#container").ejCircularGauge({
 scales: [{ height: 12 }];
 | **Property:** *axes.majorTicks.height*
 let gauge: CircularGauge = new CircularGauge({
 axes: {
 majorTicks: { height: 12 }
 };
 gauge.appendTo('#container');

|Width of Major Ticks| **Property:** *scales.ticks.width*
 \$("#container").ejCircularGauge({
 scales: [{ width: 3 }];
 | **Property:** *axes.majorTicks.width*
 let gauge: CircularGauge = new CircularGauge({
 axes: {
 majorTicks: { width: 3 }
 };
 gauge.appendTo('#container');

|Color of Major Ticks| **Property:** *scales.ticks.color*
 \$("#container").ejCircularGauge({
 scales: [{ color: "#777777" }];
 | **Property:** *axes.majorTicks.color*
 let gauge: CircularGauge = new CircularGauge({
 axes: {
 majorTicks: { color: "#777777" }
 };
 gauge.appendTo('#container');

|Offset of Major Ticks| **Property:** *scales.ticks.distanceFromScale*
 \$("#container").ejCircularGauge({
 scales: [{ distanceFromScale: 10 }];
 | **Property:** *axes.majorTicks.offset*
 let gauge: CircularGauge = new CircularGauge({
 axes: {
 majorTicks: { offset: 10 }
 };
 gauge.appendTo('#container');

|Angle of Major Ticks| **Property:** *scales.ticks.angle*
 \$("#container").ejCircularGauge({
 scales: [{ angle: 10 }];
 | Not Applicable|

|Interval of Major Ticks| **Property:** *scales.majorIntervalValue*
 \$("#container").ejCircularGauge({
 scales: [{ majorIntervalValue: 10 }];
 | **Property:** *axes.majorTicks.interval*
 let gauge: CircularGauge = new CircularGauge({
 axes: {
 majorTicks: { interval: 10 }
 };
 gauge.appendTo('#container');

|Height of Minor Ticks| **Property:** *scales.ticks.height*
 \$("#container").ejCircularGauge({
 scales: [{ type: 'minor', height: 12 }];
 | **Property:** *axes.minorTicks.height*
 let gauge: CircularGauge = new CircularGauge({
 axes: {
 minorTicks: { height: 12 }
 };
 gauge.appendTo('#container');

|Width of Minor Ticks| **Property:** *scales.ticks.width*
 \$("#container").ejCircularGauge({
 scales: [{ type: 'minor', width: 3 }];
 | **Property:** *axes.minorTicks.width*
 let gauge: CircularGauge = new CircularGauge({
 axes: {
 minorTicks: { width: 3 }
 };
 gauge.appendTo('#container');

|Color of Minor Ticks| **Property:** *scales.ticks.color*
 \$("#container").ejCircularGauge({
 scales: [{ type: 'minor', color: "#777777" }];
 | **Property:** *axes.minorTicks.color*
 let gauge: CircularGauge = new CircularGauge({
 axes: {
 minorTicks: { color: "#777777" }
 };
 gauge.appendTo('#container');

| Offset of Minor Ticks | **Property:** *scales.ticks.distanceFromScale*
 \$("#container").ejCircularGauge({ scales: [{ ticks: [{ type: 'minor', distanceFromScale: 10 }]}]; | **Property:** *axes.minorTicks.offset* let gauge: CircularGauge = new CircularGauge({ axes: [{ minorTicks: { offset: 10 } }]; gauge.appendTo('#container');

| Angle of Major Ticks | **Property:** *scales.ticks.angle* \$("#container").ejCircularGauge({ scales: [{ ticks: [{ type: 'minor', angle: 10 }]}]; | Not Applicable |

| Interval of Minor Ticks | **Property:** *scales.majorIntervalValue* \$("#container").ejCircularGauge({ scales: [{ ticks: [{ type: 'minor' }], majorIntervalValue: 10 }]; | **Property:** *axes.minorTicks.interval* let gauge: CircularGauge = new CircularGauge({ axes: [{ minorTicks: { interval: 10 } }]; gauge.appendTo('#container');

Labels

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Autoangle | **Property:** *scales.labels.autoAngle* \$("#container").ejCircularGauge({ scales: [{ labels: [{ showLabels: true, autoAngle: true }]}]; | **Property:** *axes.labelStyle.autoAngle* let gauge: CircularGauge = new CircularGauge({ axes: [{ labelStyle: { autoAngle: true } }]; gauge.appendTo('#container');

| Angle | **Property:** *scales.labels.angle* \$("#container").ejCircularGauge({ scales: [{ labels: [{ showLabels: true, angle: 30 }]}]; | Not Applicable |

| Offset | **Property:** *scales.labels.distanceFromScales* \$("#container").ejCircularGauge({ scales: [{ labels: [{ showLabels: true, distanceFromScales: 10 }]}]; | **Property:** *axes.labelStyle.offset* let gauge: CircularGauge = new CircularGauge({ axes: [{ labelStyle: { offset: 5 } }]; gauge.appendTo('#container');

| Format | **Property:** *scales.labels.unitText* \$("#container").ejCircularGauge({ scales: [{ labels: [{ unitText: "kmph", unitTextPosition: "front" }]}]; | **Property:** *axes.labelStyle.format* let gauge: CircularGauge = new CircularGauge({ axes: [{ labelStyle: { format: "kmph" } }]; gauge.appendTo('#container');

| UnitText Position | **Property:** *scales.labels.placement* \$("#container").ejCircularGauge({ scales: [{ labels: [{ showLabels: true, placement: "near" }]}]; | **Property:** *axes.labelStyle.position* let gauge: CircularGauge = new CircularGauge({ axes: [{ labelStyle: { position: "Outside" } }]; gauge.appendTo('#container');

| Label Range Color | Not Applicable | **Property:** *axes.labelStyle.useRangeColor* let gauge: CircularGauge = new CircularGauge({ axes: [{ labelStyle: { useRangeColor: true } }]; gauge.appendTo('#container');

| LabelText Color | **Property:** *scales.labels.color* \$("#container").ejCircularGauge({ scales: [{ labels: [{ color: "red" }]}]; | **Property:**

axes.labelStyle.font.color
 let gauge: CircularGauge = new CircularGauge({
 axes:
 [{ labelStyle: { font: { color: "red" } } }];
 gauge.appendTo('#container');

| Opacity | **Property:** *scales.labels.opacity*
 \$("#container").ejCircularGauge({
 scales: [{ labels: [{ opacity: 0.3 } }];
Property: *axes.labelStyle.font.opacity*
 let gauge: CircularGauge = new CircularGauge({
 axes: [{ labelStyle: { font: { opacity: 0.5 } } }];
 gauge.appendTo('#container');

| Label Font Family | **Property:** *scales.labels.font.fontFamily*
 \$("#container").ejCircularGauge({
 scales: [{ labels: [{ font: { fontFamily: "Arial" } }] }];
Property: *axes.labelStyle.font.fontFamily*
 let gauge: CircularGauge = new CircularGauge({
 axes: [{ labelStyle: { font: { fontFamily: 'Roboto' } } }];
 gauge.appendTo('#container');

| Label Font Style | **Property:** *scales.labels.font.fontStyle*
 \$("#container").ejCircularGauge({
 scales: [{ labels: [{ font: { fontStyle: "Bold" } }] }];
Property: *axes.labelStyle.font.fontStyle*
 let gauge: CircularGauge = new CircularGauge({
 axes: [{ labelStyle: { font: { fontStyle: 'Bold' } } }];
 gauge.appendTo('#container');

| Label Font Size | **Property:** *scales.labels.font.size*
 \$("#container").ejCircularGauge({
 scales: [{ labels: [{ font: { size: "12px" } }] }];
Property: *axes.labelStyle.font.size*
 let gauge: CircularGauge = new CircularGauge({
 axes: [{ labelStyle: { font: { size: '12px' } } }];
 gauge.appendTo('#container');

| Label Font Weight | Not Applicable | **Property:** *axes.labelStyle.font.fontWeight*
 let gauge:
 CircularGauge = new CircularGauge({
 axes: [{ labelStyle: { font: {
 fontWeight: 'Regular' } } }];
 gauge.appendTo('#container');

Ranges

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Start Value | **Property:** *scales.ranges.startValue*
 \$("#container").ejCircularGauge({
 scales: [{ ranges: [{ showRanges: true , startValue: 20 }] }];
Property: *axes.ranges.start*
 let gauge: CircularGauge = new CircularGauge({
 axes: [{ ranges: [{ start: 20 }] }];
 gauge.appendTo('#container');

| End Value | **Property:** *scales.ranges.endValue*
 \$("#container").ejCircularGauge({
 scales: [{ ranges: [{ showRanges: true , endValue: 30 }] }];
Property: *axes.ranges.end*
 let gauge: CircularGauge = new CircularGauge({
 axes: [{ ranges: [{ end: 30 }] }];
 gauge.appendTo('#container');

| Start Width | **Property:** *scales.ranges.startWidth*
 \$("#container").ejCircularGauge({
 scales: [{ ranges: [{ showRanges: true , startWidth: 10 }] }];
Property: *axes.ranges.startWidth*
 let gauge: CircularGauge = new CircularGauge({

 axes: [{
 ranges: [{ startWidth: 10 }]
 }] };

gauge.appendTo('#container');

| End Width | **Property:** *scales.ranges.endWidth*

 \$("#container").ejCircularGauge({

 scales: [{
 ranges: [{ showRanges: true, endWidth: 10 }]
 }] }; |
Property: *axes.ranges.endWidth*

 let gauge: CircularGauge = new CircularGauge({

 axes: [{
 ranges: [{ endWidth: 10 }]
 }] };

gauge.appendTo('#container');

| Color | **Property:** *scales.ranges.backgroundColor*

 \$("#container").ejCircularGauge({

 scales: [{
 ranges: [{ showRanges: true, backgroundColor: "red" }]

 }] }; | **Property:** *axes.ranges.color*

 let gauge: CircularGauge = new CircularGauge({

 axes: [{
 ranges: [{ color: "red" }]
 }] };

gauge.appendTo('#container');

| Offset | **Property:** *scales.ranges.distanceFromScale*

 \$("#container").ejCircularGauge({

 scales: [{
 ranges: [{ showRanges: true , distanceFromScale: 10 }]

 }] }; | Not Applicable |

| Placement | **Property:** *scales.ranges.placement*

 \$("#container").ejCircularGauge({

 scales: [{
 ranges: [{ showRanges: true, placement: "center" }]
 }] }; | Not Applicable |

| Opacity | **Property:** *scales.ranges.opacity*

 \$("#container").ejCircularGauge({

scales: [{
 ranges: [{ showRanges: true, opacity: 0.5 }]
 }] }; | Not
Applicable |

| Radius | Not Applicable | **Property:** *axes.ranges.radius*

 let gauge: CircularGauge = new
CircularGauge({
 axes: [{
 ranges: [{ radius: '80' }]
 }] };

 gauge.appendTo('#container');

| Rounded Corner Radius | Not Applicable | **Property:** *axes.ranges.roundedCornerRadius*

 let
gauge: CircularGauge = new CircularGauge({
 axes: [{
 ranges: [{
roundedCornerRadius: 10 }]
 }] };
 gauge.appendTo('#container');

| Gradients | **Property:** *scales.ranges.gradients*

 \$("#container").ejCircularGauge({

 scales: [{
 ranges: [{
 showRanges: true,

 gradients: { colorInfo: [{ colorStop : 0, color:"#FFFFFF" }] } }]
 }] }; | Not Applicable |

| Border | **Property:** *scales.ranges.border*

 \$("#container").ejCircularGauge({

scales: [{
 ranges: [{
 showRanges: true,

 border: { color: "blue", width: 2 } }]
 }] }; | Not Applicable |

Needle Pointer

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Needle Pointer | **Property:** *scales.pointers.type*

 \$("#container").ejCircularGauge({

 scales: [{
 pointers: [{ type: 'needle' }]
 }] }; | **Property:**
axes.pointers.type

 let gauge: CircularGauge = new CircularGauge({
 axes: [{

 pointers: [{ type: 'needle', value: 20 }]
 }] };

gauge.appendTo('#container');

| Needle Pointer Color| **Property:** *scales.pointers.backgroundColor*
 \$("#container").ejCircularGauge({ scales: [{ backgroundColor: 'red' }] }); | **Property:** *axes.pointers.color* let gauge: CircularGauge = new CircularGauge({ axes: [{ pointers: [{ color: 'red' }] }] }; gauge.appendTo('#container');

| Animation| **Property:** *enableAnimation*
 \$("#container").ejCircularGauge({ enableAnimation: true }); | **Property:** *axes.pointers.animation* let gauge: CircularGauge = new CircularGauge({ axes: [{ pointers: [{ animation: true, duration: 1000 }] }] }; gauge.appendTo('#container');

| Pointer Width| **Property:** *scales.pointers.width*
 \$("#container").ejCircularGauge({ scales: [{ pointers: [{ width: 5 }] }] }; | **Property:** *axes.pointers.pointerWidth* let gauge: CircularGauge = new CircularGauge({ axes: [{ pointers: [{ pointerWidth: 5 }] }] }; gauge.appendTo('#container');

| Pointer Radius| **Property:** *scales.pointers.distanceFromScale*
 \$("#container").ejCircularGauge({ scales: [{ pointers: [{ distanceFromScale: 10 }] }] }; | **Property:** *axes.pointers.radius* let gauge: CircularGauge = new CircularGauge({ axes: [{ pointers: [{ radius: 80 }] }] }; gauge.appendTo('#container');

| Opacity| **Property:** *scales.pointers.opacity*
 \$("#container").ejCircularGauge({ scales: [{ pointers: [{ opacity: 0.5 }] }] }; | Not Applicable

| Needle Type| **Property:** *scales.pointers.needleType*
 \$("#container").ejCircularGauge({ scales: [{ pointers: [{ needleType: "triangle" }] }] }; | Not Applicable

| Back Needle Length| **Property:** *scales.pointers.backNeedleLength*
 \$("#container").ejCircularGauge({ scales: [{ pointers: [{ showBackNeedle: true, backNeedleLength: 3 }] }] }; | **Property:** *axes.pointers.needleTail.length* let gauge: CircularGauge = new CircularGauge({ axes: [{ pointers: [{ needleTail: { length: 5 } }] }] }; gauge.appendTo('#container');

Marker Pointer

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Marker Pointer| **Property:** *scales.pointers.type*
 \$("#container").ejCircularGauge({ scales: [{ pointers: [{ type: 'marker' }] }] }; | **Property:** *axes.pointers.type* let gauge: CircularGauge = new CircularGauge({ axes: [{ pointers: [{ type: 'marker', value: 20 }] }] }; gauge.appendTo('#container');

| Marker Type| **Property:** *scales.pointers.markerType*
 \$("#container").ejCircularGauge({ scales: [{ pointers: [{ type: 'marker', markerType: "rectangle" }] }] }; | **Property:** *axes.pointers.markerShape* let gauge: CircularGauge = new CircularGauge({ axes: [{ pointers: [{ type: 'marker', markerShape: 'Diamond' }] }] }; gauge.appendTo('#container');

| Marker Width| **Property:** *scales.pointers.width*
 \$("#container").ejCircularGauge({
 scales: [{ type: 'marker', width: 20 }]
 });
Property: *axes.pointers.markerWidth*
 let gauge: CircularGauge = new CircularGauge({
 axes: [{ type: 'marker', markerWidth: 20 }]
 });
 gauge.appendTo('#container');

| Marker Height| **Property:** *scales.pointers.length*
 \$("#container").ejCircularGauge({
 scales: [{ type: 'marker', length: 25 }]
 });
Property: *axes.pointers.markerHeight*
 let gauge: CircularGauge = new CircularGauge({
 axes: [{ type: 'marker', markerHeight: 25 }]
 });
 gauge.appendTo('#container');

| Marker Image| **Property:** *scales.pointers.imageUrl*
 \$("#container").ejCircularGauge({
 scales: [{ type: 'marker', imageUrl: "football.png" }]
 });
Property: *axes.pointers.imageUrl*
 let gauge: CircularGauge = new
 CircularGauge({
 axes: [{ type: 'marker', imageUrl:
 "football.png" }]
 });
 gauge.appendTo('#container');

| Border Customization| **Property:** *scales.pointers.border*
 \$("#container").ejCircularGauge({
 scales: [{ type: 'marker',
 border: { color: 'red', width: 2 } }]
 });
Property:
axes.pointers.border
 let gauge: CircularGauge = new CircularGauge({
 axes: [{ type: 'marker',
 border: { color: 'red', width: 2 } }]
 });
 gauge.appendTo('#container');

Rangebar Pointer

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Rangebar| Not Applicable| **Property:** *axes.pointers.type*
 let gauge: CircularGauge = new
 CircularGauge({
 axes: [{ type: 'RangeBar' }]
 });
 gauge.appendTo('#container');

| Rounded Corner Radius| Not Applicable| **Property:** *axes.pointers.roundedCornerRadius*
 let
 gauge: CircularGauge = new CircularGauge({
 axes: [{ type: 'RangeBar', roundedCornerRadius: 10 }]
 });
 gauge.appendTo('#container');

Annotations

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Content| **Property:** *scales.customLabels.value*
 \$("#container").ejCircularGauge({
 scales: [{ value: 'Lineargauge' }]
 });
Property: *axes.annotations.content*
 let gauge: CircularGauge = new CircularGauge({
 axes: [{ content: 'Annotation' }]
 });
 gauge.appendTo('#container');

| Angle| **Property:** *scales.customLabels.textAngle*
 \$("#container").ejCircularGauge({
 scales: [{ textAngle: 90 }]
 });
Property:
axes.annotations.angle
 let gauge: CircularGauge = new CircularGauge({
 axes: [{ angle: 90 }]
 });
 gauge.appendTo('#container');

|Font Family| **Property:** *scales.customLabels.font.fontFamily*

 \$("#container").ejCircularGauge({
 scales:[{
 customLabels: [{ font: {
 fontFamily: "Arial" } }]
 }]}); | **Property:** *axes.annotations.textStyle.fontFamily*

 let
 gauge: CircularGauge = new CircularGauge({
 axes: [{
 annotations: [{
 textStyle: { fontFamily: "Arial" } }]
 }]});
 gauge.appendTo('#container'); |

|Font Color| **Property:** *scales.customLabels.color*

 \$("#container").ejCircularGauge({

 scales:[{
 customLabels: [{ color : "red" }]
 }]}); | **Property:**
axes.annotations.textStyle.color

 let gauge: CircularGauge = new CircularGauge({

 axes: [{
 annotations: [{ textStyle: { color: "red" } }]
 }]});

 gauge.appendTo('#container'); |

|Auto Angle| Not Applicable| **Property:** *axes.annotations.autoAngle*

 let gauge: CircularGauge
 = new CircularGauge({
 axes: [{
 annotations: [{ autoAngle : true }]

 }]});
 gauge.appendTo('#container'); |

|Radius| Not Applicable| **Property:** *axes.annotations.radius*

 let gauge: CircularGauge = new
 CircularGauge({
 axes: [{
 annotations: [{ radius : "10%" }]

 }]});
 gauge.appendTo('#container'); |

|Annotation Position| **Property:** *scales.customLabels.position*

 \$("#container").ejCircularGauge({
 scales:[{
 customLabels: [{ position
 : { x: 10, y: 10 } }]
 }]}); | Not Applicable |

|Annotation Position Type| **Property:** *scales.customLabels.positionType*

 \$("#container").ejCircularGauge({
 scales:[{
 customLabels: [{
 positionType : "outer" }]
 }]}); | Not Applicable |

|ZIndex| Not Applicable| **Property:** *axes.annotations.zIndex*

 let gauge: CircularGauge = new
 CircularGauge({
 axes: [{
 annotations: [{ zIndex : '1' }]
 }]});

 gauge.appendTo('#container'); |

Appearance

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

|Title| Not Applicable| **Property:** *title*

 let gauge: CircularGauge = new CircularGauge({

 title: 'Circular Gauge'
 });
 gauge.appendTo('#container'); |

|Background Color| **Property:** *backgroundColor*

 \$("#container").ejCircularGauge({

 backgroundColor : "red"
 }); | **Property:** *background*

 let gauge: CircularGauge =
 new CircularGauge({
 background : "red"
 });
 gauge.appendTo('#container'); |

|Localization| **Property:** *locale*

 \$("#container").ejCircularGauge({
 locale : "en-
 US"
 }); | **Property:** *locale*

 let gauge: CircularGauge = new CircularGauge({

 locale : "en-US"
 });
 gauge.appendTo('#container'); |

|Border| Not Applicable| **Property:** *border*

 let gauge: CircularGauge = new CircularGauge({

 border : { color: "red" , width: 2 }
 });
 gauge.appendTo('#container'); |

|Center of X| Not Applicable| **Property:** *centerX*

 let gauge: CircularGauge = new
 CircularGauge({
 centerX : "120px"
 });
 gauge.appendTo('#container'); |

|Center of Y| Not Applicable| **Property:** *centerY*
let gauge: CircularGauge = new CircularGauge({ centerY : "150px" }); gauge.appendTo('#container');

|Theme| **Property:** *theme* \$("#container").ejCircularGauge({ theme : "flatlight" }); **Property:** *theme* let gauge: CircularGauge = new CircularGauge({ theme : "Material" }); gauge.appendTo('#container');

|Margin| Not Applicable| **Property:** *margin*
let gauge: CircularGauge = new CircularGauge({ margin: { left: 40, right: 40, top: 40, bottom: 40 } }); gauge.appendTo('#container');

Events

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

|Annotation Event| **Event:** *drawCustomLabel*
\$("#container").ejCircularGauge({ drawCustomLabel: function (args) {} }); **Event:** *annotationRender*
let gauge: CircularGauge = new CircularGauge({ annotationRender: function(e: IAnnotationRenderEventArgs): void { } }); gauge.appendTo('#container');

|Label Event| **Event:** *drawLabels*
\$("#container").ejCircularGauge({ drawLabels: function (args) {} }); **Event:** *axisLabelRender*
let gauge: CircularGauge = new CircularGauge({ axisLabelRender: function(e: IAxisLabelRenderEventArgs): void { } }); gauge.appendTo('#container');

|Load Event| **Event:** *load*
\$("#container").ejCircularGauge({ load: function (args) {} }); **Event:** *load*
let gauge: CircularGauge = new CircularGauge({ load: function(e: ILoadedEventArgs): void { } }); gauge.appendTo('#container');

|Loaded Event| **Event:** *loaded*
\$("#container").ejCircularGauge({ loaded: function (args) {} }); **Event:** *loaded*
let gauge: CircularGauge = new CircularGauge({ loaded: function(e: ILoadedEventArgs): void { } }); gauge.appendTo('#container');

|Tooltip Rendered Event| Not Applicable| **Event:** *tooltipRender*
let gauge: CircularGauge = new CircularGauge({ tooltipRender: function(e: ITooltipRenderEventArgs): void { } }); gauge.appendTo('#container');

|Resized Rendered Event| Not Applicable| **Event:** *resized*
let gauge: CircularGauge = new CircularGauge({ tooltipRender: function(e: IResizeEventArgs): void { } }); gauge.appendTo('#container');

|Animation Event| Not Applicable| **Event:** *animationComplete*
let gauge: CircularGauge = new CircularGauge({ animationComplete: function(e: IAnimationCompleteEventArgs): void { } }); gauge.appendTo('#container');

|Mousedown Event| **Event:** *mouseClick*
\$("#container").ejCircularGauge({ mouseClick: function (args) {} }); **Event:** *gaugeMouseDown*
let gauge: CircularGauge = new CircularGauge({ gaugeMouseDown: function(e: IMouseEventArgs): void { } }); gauge.appendTo('#container');

|Mousemove Event| **Event:** *mouseClickMove*
\$("#container").ejCircularGauge({ mouseClickMove: function (args) {} }); **Event:** *gaugeMouseLeave*
let gauge:

```
CircularGauge = new CircularGauge({ <br/>&#160; gaugeMouseLeave: function(e: IMouseEventArgs):  
void { } <br/> }); <br/> gauge.appendTo('#container');
```

```
| Mouseup Event | Event: mouseClickUp<br/><br/> $("'#container']").ejCircularGauge({<br/>&#160;  
mouseClickUp: function (args) { } <br/> }); | Event: gaugeMouseUp<br/><br/> let gauge: CircularGauge =  
new CircularGauge({ <br/>&#160; gaugeMouseUp: function(e: IMouseEventArgs): void { } <br/> });  
<br/> gauge.appendTo('#container');
```

```
| Pointerdrag Move Event | Event: drawPointers<br/><br/>  
$("'#container']").ejCircularGauge({<br/>&#160; drawPointers: function (args) { } <br/> }); | Event:  
dragMove<br/><br/> let gauge: CircularGauge = new CircularGauge({ <br/>&#160; dragMove:  
function(e: IMouseEventArgs): void { } <br/> }); <br/> gauge.appendTo('#container');
```

```
| Draw Range Event | Event: drawRange<br/><br/> $("'#container']").ejCircularGauge({<br/>&#160;  
drawRange: function (args) { } <br/> }); | Not Applicable |
```

```
| Draw Ticks Event | Event: drawTicks<br/><br/> $("'#container']").ejCircularGauge({<br/>&#160;  
drawTicks: function (args) { } <br/> }); | Not Applicable |
```

```
| Legend Render Event | Event: legendItemRender<br/><br/>  
$("'#container']").ejCircularGauge({<br/>&#160; legendItemRender: function (args) { } <br/> }); | Not  
Applicable |
```

```
| Animation Complete Event | Not Applicable | Event: animationComplete<br/><br/> let gauge:  
CircularGauge = new CircularGauge({ <br/>&#160; animationComplete: function(e:  
IAnimationCompleteEventArgs): void { } <br/> }); <br/> gauge.appendTo('#container');
```

```
| Right Click Event | Event: rightClick<br/><br/> $("'#container']").ejCircularGauge({<br/>&#160; rightClick:  
function (args) { } <br/> }); | Not Applicable |
```

```
| Double Click Event | Event: doubleClick<br/><br/> $("'#container']").ejCircularGauge({<br/>&#160;  
doubleClick: function (args) { } <br/> }); | Not Applicable |
```

How To

Gauge range in `##Platform_Name##` Circular gauge control

Add a range to the circular gauge dynamically

You can add a range to the circular gauge dynamically by pushing the new ranges to the circular gauge axes in button click.

To add ranges dynamically to the circular gauge, follow the given steps:

Step 1:

Initialize the circular gauge with one range.

INDEX.JS

```
var circulargauge = new ej.circulargauge.CircularGauge({  
  axes: [{  
    minimum: 0, maximum: 120,  
    ranges: [  
      {  
        start: 0, end: 20  
      }  
    ]  
  }]  
});
```



```
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <button id="addRange">Add Range</button>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Step 2:

You can add ranges to the circular gauge dynamically using the button click event. In button click, add the new ranges to the circular gauge axes. To refresh the circular gauge, invoke the `refresh` method.

INDEX.JS

```
var circulargauge = new ej.circulargauge.CircularGauge({
  axes: [
    {
      minimum: 0, maximum: 120,
      ranges: [
        {
          start: 0, end: 20
        }
      ]
    }
  ]
}, '#element');
document.getElementById("addRange").onclick = function () {
```

```

    var start =
+ (circulargauge.axes[0].ranges[circulargauge.axes[0].ranges.length -
1].start) + 20;
    var end = start + 20;
    if (end > circulargauge.axes[0].maximum) {
        circulargauge.axes[0].maximum = end;
    }
    let range = { start: start, end: end };
    circulargauge.axes[0].ranges.push(new
ej.circulargauge.Range((circulargauge.axes[0].ranges[0]), 'ranges', range));
    circulargauge.refresh();
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <button id="addRange">Add Range</button>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Color Picker

Mode and value in ##Platform_Name## Color picker control

Rendering palette at initial load

By default, the **Picker** area will be rendered at initial load. To render the Palette area while opening the ColorPicker pop-up, and specify the [mode](#) property as **Palette**.

In the following sample, it will render the **Palette** at initial load.

INDEX.JS

```
ej.base.enableRipple = true;
var colorPicker = new ej.inputs.ColorPicker(
{
    //To render palette at initial load.
    mode: 'Palette'
},
'#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 ColorPicker</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript ColorPicker Component">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="wrap">
      <h4>Select color</h4>
      <input id="element" type="color">
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

STYLES.CSS

```
#container {
```

```

        visibility: hidden;
    }
    #loader {
        color: #008cff;
        font-family: 'Helvetica Neue', 'calibiri';
        font-size: 14px;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
    .wrap {
        margin: 0 auto;
        width: 300px;
        text-align: center;
    }

```

Color value

The [value](#) property can be used to specify the color value to the ColorPicker. It supports either **three** or **six** digit hex codes. To include **opacity**, set the color value as **four** or **eight** digit hex code.

In the following sample, the color value sets as **four** digit hex code, the last digit represents the **opacity** value.

INDEX.JS

```

ej.base.enableRipple = true;
var colorPicker = new ej.inputs.ColorPicker({
    //To set color value.
    value: '035a',
    mode: 'Picker',
    modeSwitcher: false
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 ColorPicker</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="TypeScript ColorPicker Component">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">

```

```
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="wrap">
            <h4>Select color</h4>
            <input id="element" type="color">
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

STYLES.CSS

```
#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    font-family: 'Helvetica Neue', 'calibri';
    font-size: 14px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.wrap {
    margin: 0 auto;
    width: 300px;
    text-align: center;
}
```

> The [value](#) property supports hex code with or without # prefix.

See Also

- [How to render palette alone](#)
- [Custom palette](#)
- [No color support in palette](#)

Localization in ##Platform_Name## Color picker control

Localization

The [Localization](#) library allows you to localize default text content of the ColorPicker. The ColorPicker component has static text for control buttons (apply / cancel) and mode switcher that can be changed to other cultures (Arabic, Deutsch, French, etc.) by defining the [locale](#) value and translation object.

The following list of properties and its values are used in the ColorPicker.

Locale key words | Text

Apply | Apply

Cancel | Cancel

ModeSwitcher | Switch Mode

Loading translations

To load translation object in an application use [load](#) function of [L10n](#) class.

The below example demonstrates the ColorPicker in Deutsch culture.

INDEX.JS

```
ej.base.enableRipple = true;
ej.base.L10n.load({
  'de-DE': {
    'colorpicker': {
      "Apply": "Anwenden",
      "Cancel": "Abbrechen",
      "ModeSwitcher": "Modus wechseln"
    }
  }
});
var colorPicker = new ej.inputs.ColorPicker({ locale: 'de-DE' },
'#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 ColorPicker</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript ColorPicker Component">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="wrap">
            <h4>Choose color</h4>
            <input id="element" type="color">
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    font-family: 'Helvetica Neue', 'calibiri';
    font-size: 14px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.wrap {
    margin: 0 auto;
    width: 300px;
    text-align: center;
}

```

Right to Left - RTL

ColorPicker component has **RTL** support. It helps to render the ColorPicker from right-to-left direction. It improves the user experiences and accessibility for users who use right-to-left languages(Arabic, Farsi, Urdu, etc). This can be achieved by setting the [enableRtl](#) property to **true**.

The following example illustrates how to enable right-to-left support in ColorPicker component.

INDEX.JS

```

ej.base.enableRipple = true;
ej.base.L10n.load({

```

```

    'ar-AE': {
      'colorpicker': {
        'Apply': 'تطبيق',
        'Cancel': 'إلغاء',
        'ModeSwitcher': 'مفتاح كهربائي الوضع'
      }
    }
  });
  var colorPicker = new ej.inputs.ColorPicker({ enableRtl: true, locale: 'ar-AE' }, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 ColorPicker</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript ColorPicker Component">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="wrap">
      <h4>Choose color</h4>
      <input id="element" type="color">
    </div>
  </div>
  <script>
  var ele = document.getElementById('container');
  if(ele) {
    ele.style.visibility = "visible";
  }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS


```
#container {
  visibility: hidden;
}
#loader {
  color: #008c9f;
  font-family: 'Helvetica Neue', 'calibiri';
  font-size: 14px;
  height: 40px;
  left: 45%;
  position: absolute;
  top: 45%;
  width: 30%;
}
.wrap {
  margin: 0 auto;
  width: 300px;
  text-align: center;
}
```

See Also

- [More information about localization](#)

Accessibility in ##Platform_Name## Color picker control

The Color picker component followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Color picker component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support | |

| [Section 508](#) Support | |

| Screen Reader Support | |

| Right-To-Left Support | |

| Color Contrast | |

| Mobile Device Support | |

| Keyboard Navigation Support | |

| [Accessibility Checker](#) Validation | |

| [Axe-core](#) Accessibility Validation | |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<div> - All features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>

WAI-ARIA attributes

The Color picker component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Color picker component:

| Attributes | Purpose |

| --- | --- |

| **role** | Indicates the Color picker component as **color** and the tiles as **gridcell** in the color palette. |

| **aria-label** | Indicates the accessible name for the tiles. |

| **aria-selected** | Indicates the current selected state of the tile. |

| **aria-haspopup** | Indicates the availability of the popup element. |

| **aria-expanded** | Indicates whether the popup can be expanded or collapsed, as well as indicates whether its current state is expanded or collapsed. |

| **aria-owns** | Identifies an elements in order to define a visual, functional, or contextual parent/child relationship between DOM elements where the DOM hierarchy cannot be used to represent the relationship. |

| **aria-disabled** | Indicates that the element is perceivable but disabled, so it is not editable or otherwise operable. |

Keyboard interaction

The Color picker component followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Color picker component.

| Press | To do this |

| --- | --- |

| **Up Arrow** | Moves the handler/tile up from the current position. |

- | Down Arrow | Moves the handler/tile down from the current position. |
- | Left Arrow | Moves the handler/tile left from the current position. |
- | Right Arrow | Moves the handler/tile right from the current position. |
- | Enter | Apply the selected color value. |
- | Tab | To focus the next focusable element in the Color picker popup. |

Ensuring accessibility

The Color picker component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Color picker component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Color picker component with accessibility tools.

See also

- [Accessibility in Syncfusion `##Platform_Name##` components](#)

Style and appearance in `##Platform_Name##` Color picker control

To modify the ColorPicker appearance, you need to override the default CSS of ColorPicker component. Please find the list of CSS classes and its corresponding section in ColorPicker component. Also, you have an option to create your own custom theme for the controls using our [Theme Studio](#).

CSS Class | Purpose of Class

- |.e-custom-picker .e-container .e-handler|To customized Color Picker selection handler
- |.color-picker.e-dropdown-popup ul .e-container|To customize the Color Picker container
- |.color-picker.e-dropdown-popup ul .e-item.e-palette-item|To customize the Color Picker pallete item
- |.color-picker.e-dropdown-popup .e-container .e-switch|To customize the Color Picker switch control
- |.color-picker.e-dropdown-popup .e-container .e-slider-preview|To customize the Color Picker slider control

How To

Hide control buttons in `##Platform_Name##` Color picker control

ColorPicker can be rendered without control buttons (Apply/Cancel). In this case, while selecting a color, the ColorPicker pop-up is closed and selected colors can be applied directly. To hide control buttons, set the [showButtons](#) property to `false`.

INDEX.JS

```
ej.base.enableRipple = true;
var colorPicker = new ej.inputs.ColorPicker({ showButtons: false },
'#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 ColorPicker</title>
```

```

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="TypeScript ColorPicker Component">
<meta name="author" content="Syncfusion">
<link href="styles.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="wrap">
            <h4>Choose color</h4>
            <input id="element" type="color">
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    font-family: 'Helvetica Neue', 'calibiri';
    font-size: 14px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.wrap {
    margin: 0 auto;
}

```

```
width: 300px;
text-align: center;
}
```

Render palette alone in `##Platform_Name##` Color picker control

To render the **Palette** alone in ColorPicker, specify the `mode` property as **Palette**, and set the `modeSwitcher` property to **false**.

In the following sample, the `showButtons` property is disabled to hide the control buttons and it renders only the **Palette** area.

INDEX.JS

```
ej.base.enableRipple = true;
var colorPicker = new ej.inputs.ColorPicker(
{
    //To render palette.
    mode: 'Palette',
    //To hide the mode switcher.
    modeSwitcher: false,
    showButtons: false
},
'#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 ColorPicker</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript ColorPicker Component">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="wrap">
      <h4>Select color</h4>
```

```

        <input id="element" type="color">
    </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008c9f;
    font-family: 'Helvetica Neue', 'calibri';
    font-size: 14px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.wrap {
    margin: 0 auto;
    width: 300px;
    text-align: center;
}

```

> To render **Picker** alone specify the [mode](#) property as 'Picker'.

Colorpicker in dropdownbutton in **##Platform_Name##** Color picker control

This section explains about how to render the ColorPicker in DropDownButton. The [target](#) property of the DropDownButton helps to achieve this scenario. To know about the usage of **target** property refer to [Popup templating](#) section.

In the below sample, the color picker is rendered as inline type by setting [inline](#) property as **true** and the rendered color picker wrapper is passed as a **target** to the DropDownButton to achieve the above scenario.

INDEX.JS

```

ej.base.enableRipple = true;
var colorPicker = new ej.inputs.ColorPicker(
    {
        inline: true,
        change: function (args) {
            ddb.element.children[0].style.backgroundColor =
args.currentValue.hex;
            closePopup();
        }
    }
);

```

```

    },
    '#element');
var ddb = new ej.splitbuttons.DropDownButton(
{
    target: ".e-colorpicker-wrapper",
    iconCss: "e-dropdownbtn-preview",
    beforeClose: function (args) {
        args.element.parentElement.querySelector('.e-cancel').removeEventListener('click', closePopup);
    },
    open: function (args) {
        args.element.parentElement.querySelector('.e-cancel').addEventListener('click', closePopup);
        tooltip();
    }
},
"#dropdownbtn");
function closePopup() {
    ddb.toggle();
}
function tooltip() {
    var zindex = (document.getElementsByClassName('e-color-picker-tooltip')[0]).style.zIndex;
    var zIndexIntValue = parseInt(zindex) + 2;
    (document.getElementsByClassName('e-color-picker-tooltip')[0]).style.zIndex = zIndexIntValue.toString();
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 ColorPicker</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="TypeScript ColorPicker Component">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

```

```
<div id="container">
  <div class="wrap">
    <h4>Choose color</h4>
    <input id="element" type="color">
    <button id="dropdownbtn"></button>
  </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

STYLES.CSS

```
#container {
  visibility: hidden;
}
#loader {
  color: #008cff;
  font-family: 'Helvetica Neue', 'calibiri';
  font-size: 14px;
  height: 40px;
  left: 45%;
  position: absolute;
  top: 45%;
  width: 30%;
}
.wrap {
  margin: 0 auto;
  width: 300px;
  text-align: center;
}
/* DropDownButton preview customization */
#dropdownbtn .e-btn-icon.e-dropdownbtn-preview {
  background-color: #008000;
  height: 18px;
  width: 18px;
  margin-top: 0;
}
#dropdownbtn {
  padding: 4px;
}
h4 {
  font-family: 'Helvetica Neue', 'Helvetica', 'Arial', 'sans-serif';
  font-size: 14px;
}
```


Customize colorpicker in `##Platform_Name##` Color picker control

Custom palette

By default, the Palette will be rendered with default colors. To load custom colors in the palette, specify the colors in the [presetColors](#) property. To customize the color palette, add a custom class to palette tiles using [beforeTileRender](#) event.

The following sample demonstrates the above functionalities.

INDEX.JS

```
ej.base.enableRipple = true;
var colorPicker = new ej.inputs.ColorPicker({
  mode: 'Palette',
  modeSwitcher: false,
  showButtons: false,
  inline: true,
  // To specify number of columns to be rendered.
  columns: 4,
  value: '#ba68c8',
  //To load custom colors.
  presetColors: {
    'custom1': ['#ef9a9a', '#e57373', '#ef5350', '#f44336', '#f48fb1',
    '#f06292',
    '#ec407a', '#e91e63', '#ce93d8', '#ba68c8', '#ab47bc',
    '#9c27b0', '#b39ddb',
    '#9575cd', '#7e57c2', '#673ab7'],
    'custom2': ['#9fa8da', '#7986cb', '#5c6bc0', '#3f51b5', '#90caf9',
    '#64b5f6',
    '#42a5f5', '#2196f3', '#81d4fa', '#4fc3f7', '#29b6f6',
    '#03a9f4',
    '#80ddea', '#4dd0e1', '#26c6da', '#00bcd4'],
    'custom3': ['#80cbc4', '#4db6ac', '#26a69a', '#009688', '#a5d6a7',
    '#81c784',
    '#66bb6a', '#4caf50', '#c5e1a5', '#aed581', '#9ccc65',
    '#8bc34a', '#e6ee9c',
    '#dce775', '#d4e157', '#cddc39'],
    'custom4': ['#fff59d', '#fff176', '#ffee58', '#ffeb3b', '#ffe082',
    '#ffd54f',
    '#ffca28', '#ffc107', '#ffcc80', '#ffb74d', '#ffa726',
    '#ff9800', '#ffab91',
    '#ff8a65', '#ff7043', '#ff5722']
  },
  beforeTileRender: function (args) {
    ej.base.addClass([args.element], ['e-icons', 'e-custom-tile']);
  },
  change: function (args) {
    document.getElementById('preview').style.backgroundColor =
    args.currentValue.rgba;
  }
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 ColorPicker</title>
  <meta charset="utf-8">
```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="TypeScript ColorPicker Component">
<meta name="author" content="Syncfusion">
<link href="styles.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="wrap">
            <div id="preview"></div>
            <h4>Select color</h4>
            <input id="element" type="color">
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

/* Default styles for the page */
#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    font-family: 'Helvetica Neue', 'calibiri';
    font-size: 14px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.wrap {

```

```

margin: 0 auto;
width: 300px;
text-align: center;
}
#preview {
background-color: #ba68c8;
height: 50px;
width: 100%;
}
/* Tile customization styles */
#element+.e-container {
background-color: transparent;
border-color: transparent;
box-shadow: none;
}
#element+.e-container .e-custom-palette.e-palette-group {
height: 182px;
}
#element+.e-container .e-palette .e-custom-tile {
border: 0;
color: #fff;
height: 36px;
font-size: 18px;
width: 36px;
line-height: 36px;
border-radius: 50%;
margin: 2px 5px;
}
/* Selected state icon */
#element+.e-container .e-palette .e-custom-tile.e-selected::before {
content: '\e933';
}
#element+.e-container .e-palette .e-custom-tile.e-selected {
outline: none;
}

```

Hide input area from picker

By default, the input area will be rendered in ColorPicker. To hide the input area from it, add `e-hide-value` class to ColorPicker using the [cssClass](#) property.

In the following sample, the ColorPicker is rendered without input area.

INDEX.JS

```

ej.base.enableRipple = true;
var colorPicker = new ej.inputs.ColorPicker({ cssClass: 'e-hide-value' },
'#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
<title>EJ2 ColorPicker</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="TypeScript ColorPicker Component">
<meta name="author" content="Syncfusion">

```

```

<link href="styles.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="wrap">
            <h4>Choose color</h4>
            <input id="element" type="color">
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    font-family: 'Helvetica Neue', 'calibiri';
    font-size: 14px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.wrap {
    margin: 0 auto;
    width: 300px;
    text-align: center;
}

```

Custom handle

Color picker handle shape and UI can be customized. Here, we have customized the handle as **svg icon**. The same way you can customize the handle based on your requirement.

The following sample show the customized color picker handle.

INDEX.JS

```
ej.base.enableRipple = true;
var colorPicker = new ej.inputs.ColorPicker(
  {
    value: '#344aee',
    // To hide the input area
    cssClass: 'e-custom-picker',
    modeSwitcher: false,
    open: function(args) {
      args.element.querySelector('.e-handler').classList.add('e-
icons');
    }
  }, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 ColorPicker</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript ColorPicker Component">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="wrap">
      <h4>Choose color</h4>
      <input id="element" type="color">
    </div>
  </div>
</body>
</html>
```

```
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

STYLES.CSS

```

/* Default styles for the page */
#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    font-family: 'Helvetica Neue', 'calibiri';
    font-size: 14px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.wrap {
    margin: 0 auto;
    width: 300px;
    text-align: center;
}
/* To hide the handle balloon preview */
.e-color-picker-tooltip.e-popup.e-popup-open {
    display: none;
}
/* Handle customization styles */
.e-custom-picker .e-container .e-hsv-container .e-handler {
    background: transparent
url('data:image/svg+xml;base64,PD94bWwgdGVyc2lvbj0iMS4wIiBlbmNvZGluc2VudD0idXRmL
TgiPz4KPCFtLSBHZW51cmF0b3I6IEFkb2JlIElsbHVzdHJhdG9yIDYyLjEuMCwgU1ZHIIEV4cG9yd
CBQbHVnLULuIC4gU1ZHFZlcnNpb246IDYuMDAgQnVpbGQgMCkgIC0tPgo8c3ZnIHZlcnNpb249I
jEuMSIgaWQ9IkkxeWVyXzEiIHhtbG5zPSJodHRwOi8vd3d3LnczLm9yZy8yMDAwL3N2ZyIgeGlsb
nM6eGxpbnM9Imh0dHA6Ly93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2Zy
IgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3
Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3
N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93
d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMD
AwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGls
bG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy
8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyI
geGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3
Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3
N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93
d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMD
AwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGls
bG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy
8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyI
geGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3
Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3
N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93
d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMD
AwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGls
bG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy
8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyI
geGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3
Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3
N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93
d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMD
AwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGls
bG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy
8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyI
geGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3
Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3
N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93
d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMD
AwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGls
bG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy
8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyI
geGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3
Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3
N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93
d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMD
AwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGls
bG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy
8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyI
geGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3
Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3
N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93
d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMD
AwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGls
bG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy
8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyI
geGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3
Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3
N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93
d3d3Lm9yZy8yMDAwL3N2ZyIgeGlsbG93d3d3Lm
```

```
h4 {  
    font-family: 'Helvetica Neue', 'Helvetica', 'Arial', 'sans-serif';  
    font-size: 14px;  
}
```

Custom primary button

By default, the applied color will be updated in primary button of the color picker. You can customize that as **icon**.

In the following sample, the **picker** icon is added to primary button and using [change](#) event the selected color will be updated in bottom portion of the icon.

INDEX.JS

```
ej.base.enableRipple = true;  
var colorPicker = new ej.inputs.ColorPicker(  
    {  
        change: function (args) {  
  
            colorPicker.element.nextElementSibling.querySelector('.e-selected-color').style.borderColor = args.currentValue.rgba;  
        }  
    },  
    '#element');  
ej.base.addClass([colorPicker.element.nextElementSibling.querySelector('.e-selected-color')], 'e-icons');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>  
    <title>EJ2 ColorPicker</title>  
    <meta charset="utf-8">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <meta name="description" content="TypeScript ColorPicker Component">  
    <meta name="author" content="Syncfusion">  
    <link href="styles.css" rel="stylesheet">  
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">  
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">  
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">  
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">  
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">  
  
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js" type="text/javascript"></script>  
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type="text/javascript"></script>  
</head>  
<body>
```

```

<div id="container">
  <div class="wrap">
    <h4>Choose color</h4>
    <input id="element" type="color">
  </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
  visibility: hidden;
}
#loader {
  color: #008cff;
  font-family: 'Helvetica Neue', 'calibiri';
  font-size: 14px;
  height: 40px;
  left: 45%;
  position: absolute;
  top: 45%;
  width: 30%;
}
.wrap {
  margin: 0 auto;
  width: 300px;
  text-align: center;
}
/* Icon customization */
.e-colorpicker-wrapper #element+.e-split-btn-wrapper .e-split-btn .e-
selected-color {
  background: none;
  border-bottom-style: solid;
  border-bottom-width: 3px;
  width: 14px;
  margin: 0px 2px;
  border-bottom-color: #008000;
}
.e-colorpicker-wrapper #element+.e-split-btn-wrapper .e-split-btn .e-
selected-color .e-split-preview {
  display: none;
}
.e-colorpicker-wrapper #element+.e-split-btn-wrapper .e-split-btn .e-
selected-color::before {
  content: '\e35c';
}
h4 {
  font-family: 'Helvetica Neue', 'Helvetica', 'Arial', 'sans-serif';
  font-size: 14px;
}

```



```
}

```

> The Essential JS 2 provides a set of icons that can be loaded by applying `e-icons` class name to the element. You can also use third party icon to customize the primary button.

Display hex code in input

The color picker input element can be showcased in the place of primary button. The applied color hex code will be updated in the primary button input.

The following sample shows the color picker with input.

INDEX.JS

```
ej.base.enableRipple = true;
var colorPicker = new ej.inputs.ColorPicker({}, '#element');
var target = colorPicker.element.nextElementSibling;
target.insertBefore(colorPicker.element, target.children[1]);
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 ColorPicker</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript ColorPicker Component">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="wrap">
      <h4>Choose color</h4>
      <input id="element" class="e-input" type="text" readonly="">
    </div>
  </div>
</body>
</html>

<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
```

```

}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    font-family: 'Helvetica Neue', 'calibiri';
    font-size: 14px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.wrap {
    margin: 0 auto;
    width: 300px;
    text-align: center;
}
/* Input element customization */
.e-colorpicker-wrapper .e-split-btn-wrapper #element.e-input {
    height: 16px;
    margin: 0;
    opacity: 1;
    position: initial;
    width: 75px;
}
/* To hide primary button */
.e-colorpicker-wrapper .e-split-btn-wrapper .e-split-btn {
    display: none;
}
/* Secondary button customization */
.e-colorpicker-wrapper .e-split-btn-wrapper .e-btn.e-dropdown-btn {
    background: transparent;
    border-color: transparent;
    border-bottom-color: rgba(0, 0, 0, 0.42);
}
.e-colorpicker-wrapper .e-split-btn-wrapper .e-input:focus+.e-btn.e-
dropdown-btn {
    padding-bottom: 3px;
    border-bottom-width: 2px;
    border-bottom-color: #e3165b;
}
.e-colorpicker-wrapper .e-split-btn-wrapper .e-btn.e-dropdown-btn .e-caret {
    transform: rotate(0deg);
    transition: transform 200ms ease-in-out;
}
.e-colorpicker-wrapper .e-split-btn-wrapper .e-btn.e-dropdown-btn.e-active
.e-caret {
    transform: rotate(180deg);
}

```

```

}
h4 {
  font-family: 'Helvetica Neue', 'Helvetica', 'Arial', 'sans-serif';
  font-size: 14px;
}

```

Custom UI

The color picker UI can be customized in all possible ways. The following sample shows the excel like UI customization with help of SplitButton and Dialog component. In that by clicking the more colors option from color palette, the dialog contains color picker will open.

INDEX.JS

```

ej.base.enableRipple = true;
var colorPalette = new ej.inputs.ColorPicker(
  {
    mode: 'Palette',
    inline: true,
    showButtons: false,
    modeSwitcher: false,
    change: paletteOnChange
  }
, '#palette');
var splitBtn = new ej.splitbuttons.SplitButton(
  {
    target: '#target',
    iconCss: 'e-icons e-font-icon',
    open: function (args) {
      args.element.children[1].addEventListener('click', openDialog);
    },
    beforeClose: function (args) {
      args.element.children[1].removeEventListener('click',
openDialog);
    }
  }
, '#split-btn');
var modalDialog = new ej.popups.Dialog(
  {
    target: '.wrap',
    width: '270px',
    height: '336px',
    isModal: true,
    visible: false,
    cssClass: 'e-dlg-picker',
    content: '<input type="color" id="picker" />',
    animationSettings: { effect: 'Zoom' },
    open: dialogOpen,
    overlayClick: dlgClose
  }
, '#modal-dialog');
var colorPicker = new ej.inputs.ColorPicker(
  {
    inline: true,
    modeSwitcher: false,
    change: pickerOnChange
  }

```

```

, '#picker');
function openDialog() {
    modalDialog.show();
}
function dialogOpen() {
    colorPicker.refresh();
    colorPicker.element.nextElementSibling.querySelector('.e-ctrl-btn .e-cancel').addEventListener('click', dlgClose);
}
function dlgClose() {
    modalDialog.hide();
}
function paletteOnChange(args) {
    splitBtn.element.querySelector('.e-font-icon').style.borderBottomColor = args.currentValue.rgba;
}
function pickerOnChange(args) {
    paletteOnChange(args);
    dlgClose();
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 ColorPicker</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript ColorPicker Component">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="wrap">
      <ul id="target" tabindex="0">
        <li class="e-item e-palette-item">
          <input id="palette" type="color">
        </li>
        <li class="e-item" tabindex="-1">

```

```

        <span class="e-menu-icon"></span>
        More colors...
    </li>
</ul>
<h4>Select color</h4>
<button id="split-btn"></button>
<div id="modal-dialog"></div>
</div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    font-family: 'Helvetica Neue', 'calibri';
    font-size: 14px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.wrap {
    margin: 0 auto;
    width: 100%;
    height: 100%;
    min-height: 350px;
    text-align: center;
}
/* Primary button icon preview */
.e-btn-icon.e-font-icon {
    border-bottom-style: solid;
    border-bottom-width: 3px;
}
/* Primary button icon */
.e-btn-icon.e-font-icon::before {
    content: '\e34c';
}
.e-dropdown-popup ul .e-item:first-child.e-palette-item {
    height: auto;
    padding: 0;
}
.e-dlg-picker.e-dialog .e-dlg-content {
    padding: 0;
    background-color: transparent;
}

```

```

}
/* Sets ColorPicker height */
.e-dlg-picker.e-dialog {
  max-height: 336px !important;
}
/* More colors li icon customization */
.e-dropdown-popup ul .e-item:last-child .e-menu-icon {
  height: 24px;
  margin-top: 6px;
  width: 24px;
  background-image: linear-gradient(to bottom, #fff 0, #000 100%);
  background-color: #0450c2;
  background-blend-mode: hard-light;
}
h4 {
  font-family: 'Helvetica Neue', 'Helvetica', 'Arial', 'sans-serif';
  font-size: 14px;
}

```

Handle no color support in ##Platform_Name## Color picker control

The ColorPicker component supports no color functionality. By clicking the no color tile from palette, the selected color becomes **empty** and considered as no color has been selected from color picker.

Default no color

To achieve this, set [noColor](#) property as **true**.

In the following sample, the first tile of the color palette represents the no color tile. By clicking the no color tile you can achieve the above functionalities.

INDEX.JS

```

ej.base.enableRipple = true;
var preview = document.getElementById('preview');
var colorPicker = new ej.inputs.ColorPicker(
  {
    mode: "Palette",
    value: "#ba68c8",
    showButtons: false,
    modeSwitcher: false,
    //To enable no color support
    noColor: true,
    change: function(args) {
      preview.style.backgroundColor = args.currentValue.hex;
      preview.textContent = args.currentValue.hex ?
args.currentValue.hex : 'No color';
    }
  },
  '#element');
preview.style.backgroundColor = "#ba68c8";
preview.textContent = "#ba68c8";

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 ColorPicker</title>

```

```

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="TypeScript ColorPicker Component">
<meta name="author" content="Syncfusion">
<link href="styles.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="wrap">
            <div id="preview"></div>
            <h4>Select color</h4>
            <input id="element" type="color">
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    font-family: 'Helvetica Neue', 'calibiri';
    font-size: 14px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.wrap {

```

```

margin: 0 auto;
width: 300px;
text-align: center;
}
#preview {
border: 1px solid;
height: 40px;
line-height: 40px;
}
h4, #preview {
font-family: 'Helvetica Neue', 'Helvetica', 'Arial', 'sans-serif';
font-size: 14px;
}

```

If the [noColor](#) property is enabled, make sure to disable the [modeswitcher](#) property.

Custom no color

The following sample show the color palette with custom no color option.

INDEX.JS

```

ej.base.enableRipple = true;
var preview = document.getElementById('preview');
var colorPicker = new ej.inputs.ColorPicker(
{
    mode: "Palette",
    value: "#f44336",
    showButtons: false,
    modeSwitcher: false,
    inline: true,
    columns: 4,
    presetColors: { 'custom': ['#f44336', '#e91e63', '#9c27b0',
'#673ab7', '#2196f3', '#03a9f4', '#00bcd4',
'#009688', '#8bc34a', '#cddc39', '#ffeb3b', '#ffc107'] },
    beforeTileRender: function(args) {
        args.element.classList.add('e-custom-tile');
    },
    change: function (args) {
        document.querySelector(".e-split-btn .e-picker-
icon").style.borderBottomColor = args.currentValue.hex;
        preview.style.backgroundColor = args.currentValue.hex;
        preview.textContent = args.currentValue.hex;
        if (splitBtn.element.getAttribute("aria-expanded")) {
            splitBtn.toggle();
            splitBtn.element.focus();
        }
    },
    '#element');
var splitBtn = new ej.splitbuttons.SplitButton({ iconCss: "e-cp-icons e-
picker-icon", target: "#target" }, '#splitbtn')
preview.style.backgroundColor = "#f44336";
preview.textContent = "#f44336";
document.getElementById('no-color').onclick = function() {
    //sets color picker value property to null
    colorPicker.setProperties({ 'value': "" }, true);
}

```



```

        document.querySelector('.e-split-btn .e-picker-
        icon').style.borderBottomColor = "transparent";
        preview.textContent = "No color"
        preview.style.backgroundColor = "transparent";
    }

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 ColorPicker</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="TypeScript ColorPicker Component">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
    base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
    buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
    popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
    splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
    inputs/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="wrap">
            <ul id="target" tabindex="0">
                <li class="e-item e-palette-item">
                    <input id="element" type="color">
                </li>
                <li class="e-item" id="no-color" tabindex="-1">
                    <span class="e-menu-icon e-nocolor"></span>
                    No color
                </li>
            </ul>
            <div>
                <div id="preview"></div>
                <h4>Select color</h4>
                <button id="splitbtn"></button>
            </div>
        </div>
    </div>

    <script>
    var ele = document.getElementById('container');
    if(ele) {
        ele.style.visibility = "visible";
    }

```

```

}
    </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```
#container {
    visibility: hidden;
}
#loader {
    color: #008c00;
    font-family: 'Helvetica Neue', 'calibri';
    font-size: 14px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
@font-face {
    font-family: 'paint';
    src:
        url(data:application/x-font-ttf;charset=utf-8;base64,AAEAAAAKAIAAAwAgTlMvMj0gSRIAAAEoAAAAVmntYXDnEodVAAABiAAAADZnbHlmIzD
+uwAAACgAAADMaGVhZBKhHQAADQAAAAANmhoZWEHjANrAAAArAAAACRobXR4B+j/8wAAAYAAAAA
IbG9jYQBMAAAAAHAHAABmlheHABDgBKAAABCAAAACBuYw1ln6hzswAAApQAAAINcG9zdEkLMmU
AAASkAAAAANGABAAADUv9qAFoEAP/z//4D6gABAAAAAAAAAAAAAAAAAAAAAgABAAAAAQAAAZfc6F8
PPPUACwPoAAAAANfSn9kAAAAA19Kf2f/z//wD6gPhAAAAACAACAAAAAAAAAAAAEAAAACAD4AAgAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAAAQP0AZAABQAAANoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnAANS/2oAWgPhAJYAAAABAAAAAAAAABAAAAAPo//M
AAAAACAAAAAwAAABQAAwABAAAAFAAEACIAAAAEAAQAAQAA5wD//wAA5wD//wAAAAEABAAAAAEAAAA
AAAAAZgAAAAAL/8//8A+oD4QAKAD0AAAEWBgceATc1JiQHJTMmNjceARcVJx4BBx4BFQ4BIiYnNDY
3PgEvAS4BIw4BBwEGHgI3AT4BLwE1LgEnDgEDeIrLCgulCxp+8RT+GyYDQFxoZQwTBQEDDxEBJzo
nAREOCQkPJQ4cDbcdAf6oG1a3nx8BWQ4RHKADEg1oWwHTLHVwYVmL6Kx1BHEqfwYFqWUHEX4tDAo
cEx0nJx0RHGoVUDQpDgsBFAH+px2guFUaAvkNoiCgCXnhCAW0AAAAAAAAAEgDeAAEAAAAAAAAAAQA
AAAEAAAAAAAAAEABQABAAEAAAAAAAAAIAbwAGAAEAAAAAAAAAMABQANAAEAAAAAAAAAQABQASAAEAAAAAAAU
ACwAXAAEAAAAAAAAAYABQAIAAEAAAAAAAAAoALAAEAAAAAAAAAsAEgBTAAMAAQQJAAAAAgBLaAMAAQQ
JAAEACgBnAAMAAQQJAAIADgBxAAMAAQQJAAMACgB/AAMAAQQJAAQACgCJAAMAAQQJAAUAFgCTAAM
AAQQJAAAYACgCpAAMAAQQJAAoAWACzAAMAAQQJAAsAJAELIHBhAW50UmvndWxhcnBhAW50cGFpbmR
WZXJzaW9uIDEuMHBhAW50Rm9udCBnZW51cmF0ZWQgdXNpbmcgU3luY2Zlc2lubiBNZXRybyBtdHV
kaW93d3cuc3luY2Zlc2lubi5jb20AIAbwAGEAaQBwAHQAUGBlAGcAdQBSAGEAcgBwAGEAaQBwAHQ
AcABhAGkAbgB0AFYAZQByAHMAaQBvAG4AIAAxAAC4AMABwAGEAaQBwAHQAQARgBvAG4AdAAgAGcAZQB
uAGUAcgBhAHQAQZQBkACAAdQBzAGkAbgBnACAAUwB5AG4AYwBmAHUAcwBpAG8ABgAgAE0AZQB0AHI
AbwAgAFMAcABlAGQAaQBvAHcAdwB3AC4AcwB5AG4AYwBmAHUAcwBpAG8ABgAgAGMAbwBtAAAAAAI
AAAAAAAAACgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAgECAQMADHBhAW50LWJlY2tldAAAAA=)
format('truetype');
    font-weight: normal;
    font-style: normal;
}
.e-cp-icons {
    font-family: 'paint' !important;
    speak: none;
    font-size: 55px;
    font-style: normal;
    font-weight: normal;
```

```
font-variant: normal;
text-transform: none;
line-height: 1;
-webkit-font-smoothing: antialiased;
-moz-osx-font-smoothing: grayscale;
}
/* Preview area styles */
#preview {
    border: 1px solid;
    height: 40px;
    line-height: 40px;
    width: 100%;
}
.wrap {
    margin: 0 auto;
    width: 300px;
    text-align: center;
}
/* ColorPicker customization */
.e-dropdown-popup ul#target {
    padding: 0;
}
.e-dropdown-popup ul .e-item.e-palette-item {
    height: auto;
    padding: 0;
}
.e-btn-icon.e-picker-icon {
    border-bottom-color: #f44336;
    border-bottom-style: solid;
    border-bottom-width: 3px;
}
/* Picker icon */
.e-btn-icon.e-picker-icon::before {
    content: '\e700';
}
/* No color li styles */
.e-dropdown-popup ul .e-item .e-menu-icon.e-nocolor {
    height: 22px;
    margin-top: 8px;
    width: 22px;
    background: transparent
url('data:image/svg+xml;base64,PD94bWwgdGVyc2lvbj0iMS4wIiBlbmNvZGlucz0iVVRGLTgiPz4KPHN2YyB3aWR0aD0iNnB4IiBoZWlnaHQ9IjZweCIgdmllld0JveD0iMCAwIDYgNiIgdmVyc2lvbj0iMS4xIiB4bWxuczc0iaHR0cDovL3d3dy53My5vcmcvMjAwMC9zdmciaHhtbG5zOnhsaW5rPSJodHRwOi8vd3d3LnczLm9yZy8xOTk5L3hsaW5rIj4KICAgIDwhLS0gr2VuZXJhdG9yOiBTa2V0Y2ggNTAgKDU0OTgzKSAtIGh0dHA6Ly93d3cuYm9oZWlpYW5jb2RpbmcuY29tL3NrZXRjaCatLT4KI CAgidX0aXRszT5Hcm9lcA5PC90aXRszT4KICAgIDxkZXNjPkNyZWFOZWQgd2l0aCBTa2V0Y2guPC9kZXNjPgogICAgPGRlZnM+PC9kZWZzPgogICAgPGcgawQ9IlBhZ2UtMSIgc3Ryb2t1PSJub251I iBzdHJva2Utd2lkdgG9IjEiIGZpbGw9Im5vbmluIGZpbGwtcnVsZT0iZXZlbm9kZCI+CiAgICAgIC AgPGcgawQ9Ikdyb3VwLTkiPgogICAgICAgICAgICA8cmVjdCBpZD0iUmVjdGFuZ2xlLTExIiBma WxsPSIjRTBFMEUwIiB4PSIwIiB5PSIwIiB3aWR0aD0iMyIGAqVpZ2h0PSIzIj48L3JlY3Q+CIAgI CAgICAgICAgIDxyZWN0IGlkPSJSZWN0YW5nbGUtMTETq29weS0yIiBmaWxsPSIjRkZGRkZGIiB4P SIwIiB5PSIzIiB3aWR0aD0iMyIGAqVpZ2h0PSIzIj48L3JlY3Q+CIAgICAgICAgICAgIDxyZWN0I GlkPSJSZWN0YW5nbGUtMTETq29weSIgZmlsbD0iIOZGRkZGRiIgeD0iMyIgeT0iMCIgd2lkdgG9I jMiIGHlaWdobD0iMyI+PC9yZWN0PgogICAgICAgICAgICA8cmVjdCBpZD0iUmVjdGFuZ2xlLTExL UNvcHktMyIGAqZmlsbD0iIOUwRTBFMCigeD0iMyIgeT0iMyIgeT0iMyIgd2lkdgG9IjMiIGHlaWdobD0iMyI+ PC9yZWN0PgogICAgICAgIDwvZz4KICAgIDwvZz4KCPC9zdmc+');

```

```

}
/* Tile customization */
.e-container .e-palette .e-tile.e-custom-tile {
  height: 24px;
  width: 24px;
  margin: 4px;
}
h4, #preview {
  font-family: 'Helvetica Neue', 'Helvetica', 'Arial', 'sans-serif';
  font-size: 14px;
}

```

Disabled in ##Platform_Name## Color picker control

To achieve disabled state in ColorPicker, set the [disabled](#) property to `true`. The ColorPicker pop-up cannot be accessed in disabled state.

The following example shows the `disabled` state of ColorPicker component.

INDEX.JS

```

ej.base.enableRipple = true;
var colorPicker = new ej.inputs.ColorPicker({ disabled: true }, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 ColorPicker</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript ColorPicker Component">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="wrap">
      <h4>Disabled state</h4>
      <input id="element" type="color">

```

```

        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008c8f;
    font-family: 'Helvetica Neue', 'calibri';
    font-size: 14px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.wrap {
    margin: 0 auto;
    padding-top: 30px;
    width: 300px;
    text-align: center;
}

```

ComboBox

Tags in ##Platform_Name## Combo box control

The ComboBox can be initialized on three different tags as described in below. Though it is initialized in different tags, the UI appearance and built-in features behave in the same way.

Select element

When a ComboBox is initialized on SELECT element, the list items can be assigned through the option tag of the HTML select element.

- The nested items are wrapped and grouped based on the tag that is available within the `



INDEX.JS

```

// initialize ComboBox component
let comboBoxObject = new ej.dropdowns.ComboBox({
    placeholder: "Select a vegetable",
    popupHeight: "200px"
});

```

```
// render initialized ComboBox
comboBoxObject.appendTo('#selectElement');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 ComboBox</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:50px auto 0; width:250px;">
    <br>
    <select id="selectElement">
      <optgroup label="Beans">
        <option value="1">Chickpea</option>
        <option value="2">Green bean</option>
        <option value="3" selected="selected">Horse gram</option>
      </optgroup>
      <optgroup label="Leafy and Salad">
        <option value="5">Cabbage</option>
        <option value="4">Spinach</option>
        <option value="6">Wheat grass</option>
        <option value="7">Yarrow</option>
      </optgroup>
    </select>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

UL element

The ComboBox can be initialized through `` element which contains a collection of `` element. The `` items act as a popup list items of the ComboBox. The inner text of the `` element is considered both as text and value fields.

INDEX.JS

```
//initiates the component
let comboBoxObject = new ej.dropdowns.ComboBox({
  placeholder: "Select a vegetable"
});
// render initialized ComboBox
comboBoxObject.appendTo('#ulElement');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 ComboBox</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:50px auto 0; width:250px;">
    <br>
    <ul id="ulElement">
      <li>Badminton</li>
      <li>Cricket</li>
      <li>Football</li>
      <li>Golf</li>
    </ul>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Input element

The ComboBox has also be rendered through `<input>` element with an array of either simple or complex data that is set through the [dataSource](#) property. It can retrieve data from local data sources as well as remote data services.

Detailed information about the data binding with an example is available in: [Data Binding to ComboBox](#)

Data binding in ##Platform_Name## Combo box control

The ComboBox loads the data either from local data sources or remote data services using the [dataSource](#) property. It supports the data type of array or `DataManager`.

The ComboBox also supports different kinds of data services such as OData, OData V4, and Web API, and data formats such as XML, JSON, and JSONP with the help of `DataManager` adaptors.

Fields	Type	Description
text	string	Specifies the display text of each list item.
value	number or string	Specifies the hidden data value mapped to each list item that should contain a unique value.
groupBy	string	Specifies the category under which the list item has to be grouped.
iconCss	string	Specifies the icon class of each list item.

When binding complex data to the ComboBox, fields should be mapped correctly. Otherwise, the selected item remains undefined.

Binding local data

Local data can be represented in two ways as described below.

1. Array of simple data

The ComboBox has support to load array of primitive data such as strings and numbers. Here, both value and text field act the same.

INDEX.JS

```
var sportsData = ['Badminton', 'Cricket', 'Football', 'Golf', 'Tennis'];
// initialize ComboBox component
var listObj = new ej.dropdowns.ComboBox({
  dataSource: sportsData,
  // set placeholder to ComboBox input element
  placeholder: "Select games"});
listObj.appendTo('#comboelement');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 ComboBox</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
```



```
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin:50px auto 0; width:250px;">
        <br>
        <input type="text" tabindex="1" id="comboelement">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

2. Array of JSON data

The ComboBox can generate its list items through an array of complex data. For this, the appropriate columns should be mapped to the [fields](#) property.

In the following example, **Id** column and **Game** column from complex data have been mapped to the **value** field and **text** field, respectively.

INDEX.JS

```
let sportsData = [
    { Id: 'Game1', Game: 'Badminton' },
    { Id: 'Game2', Game: 'Basketball' },
    { Id: 'Game3', Game: 'Cricket' },
    { Id: 'Game4', Game: 'Football' },
    { Id: 'Game5', Game: 'Golf' },
    { Id: 'Game6', Game: 'Hockey' },
    { Id: 'Game7', Game: 'Rugby' },
    { Id: 'Game8', Game: 'Snooker' }
];

// initialize ComboBox component
var listObj = new ej.dropdowns.ComboBox({
    //set the data to dataSource property
    dataSource: sportsData,
    // maps the appropriate column to fields property
    fields: { value: 'Game' },
    // set placeholder to ComboBox input element
    placeholder: "Find a game"});
listObj.appendTo('#comboelement');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 ComboBox</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:50px auto 0; width:250px;">
    <br>
    <input type="text" id="comboelement">
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

3. Array of Complex data

The ComboBox can generate its list items through an array of complex data. For this, the appropriate columns should be mapped to the [fields](#) property.

In the following example, **Code.Id** column and **Country.Name** column from complex data have been mapped to the **value** field and **text** field, respectively.

INDEX.JS

```
let countriesData = [
  { Country: { Name: 'Australia' }, Code: { Id: 'AU' } },
  { Country: { Name: 'Bermuda' }, Code: { Id: 'BM' } },
  { Country: { Name: 'Canada' }, Code: { Id: 'CA' } },
  { Country: { Name: 'Cameroon' }, Code: { Id: 'CM' } },
  { Country: { Name: 'Denmark' }, Code: { Id: 'DK' } },
  { Country: { Name: 'France' }, Code: { Id: 'FR' } }
];
```

```
// initialize ComboBox component
var listObj = new ej.dropdowns.ComboBox({
  //set the data to dataSource property
  dataSource: countriesData,
  // maps the appropriate column to fields property
  fields: { value: 'Country.Name' },
  // set placeholder to ComboBox input element
  placeholder: "Find a country"});
listObj.appendTo('#comboelement');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 ComboBox</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:50px auto 0; width:250px;">
    <br>
    <input type="text" id="comboelement">
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Binding remote data

The ComboBox supports retrieval of data from remote data services with the help of **DataManager** component. The **Query** property is used to fetch data from the database and bind it to the ComboBox. In the following sample, displayed first 6 contacts from the **customer** table of **Northwind** Data Service.

INDEX.JS

```
//initiates the component
var comboBoxObject = new ej.dropdowns.ComboBox({
  //bind the data manager instance to dataSource property
  dataSource: new ej.data.DataManager({
    url: 'https://services.odata.org/V4/Northwind/Northwind.svc/',
    adaptor: new ej.data.ODataV4Adaptor(),
    crossDomain: true
  }),
  //bind the Query instance to query property
  query: new ej.data.Query().from('Employees').select(['FirstName',
'City','EmployeeID']).take(6),
  //map the appropriate columns to fields property
  fields: { value: 'FirstName' },
  //set the placeholder to ComboBox input
  placeholder: "Select an employee",
  //sort the resulted items
  sortOrder: 'Ascending'
});
//render the component
comboBoxObject.appendTo('#comboelement');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 ComboBox</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:50px auto 0; width:250px;">
    <br>
    <input type="text" id="comboelement">
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
<script src="index.js" type="text/javascript"></script>
```

```
</body></html>
```

See Also

- [How to achieve cascading](#)
- [How to load data using template](#)
- [How to group the data using header](#)
- [How to filter the bound data](#)

Templates in ##Platform_Name## Combo box control

The ComboBox has been provided with several options to customize each list item, group title, selected value, header, and footer elements. It uses the Essential JS 2 [Template engine](#) to compile and render the elements properly.

Item template

The content of each list item within the ComboBox can be customized with the help of [itemTemplate](#) property.

In the following sample, each list item is split into two columns to display relevant data's.

INDEX.JS

```
//initiates the component
var comboBoxObject = new ej.dropdowns.ComboBox({
  //bind the data manager instance to dataSource property
  dataSource: new ej.data.DataManager({
    url: 'https://services.odata.org/V4/Northwind/Northwind.svc/',
    adaptor: new ej.data.ODataV4Adaptor(),
    crossDomain: true
  }),
  //bind the Query instance to query property
  query: new ej.data.Query().from('Employees').select(['FirstName',
'City', 'EmployeeID']).take(6),
  //map the appropriate columns to fields property
  fields: { value: 'FirstName' },
  //set the placeholder to ComboBox input
  placeholder: "Select an employee",
  //sort the resulted items
  sortOrder: 'Ascending',
  //set the value to itemTemplate property
  itemTemplate: "<span><span class='name'>${FirstName}</span><span class
='city'>${City}</span></span>"
});
//render the component
comboBoxObject.appendTo('#comboelement');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 ComboBox</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
```

```

<link href="styles.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin:50px auto 0; width:250px;">
        <input type="text" id="comboelement">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Group template

The group header title under which appropriate sub-items are categorized can also be customize with the help of [groupTemplate](#) property. This template is common for both inline and floating group header template.

In the following sample, employees are grouped according to their city.

INDEX.JS

```

var groupPredicate = new ej.data.Predicate('City',
'equal','london').or('City','equal','seattle');
//initiates the component
var comboBoxObject = new ej.dropdowns.ComboBox({
    //bind the data manager instance to dataSource property
    dataSource: new ej.data.DataManager({
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/',
        adaptor: new ej.data.ODataV4Adaptor(),
        crossDomain: true
    }),
    //bind the Query instance to query property
    query: new ej.data.Query().from('Employees').select(['FirstName',
'City','EmployeeID']).take(6).where(groupPredicate),
    //map the appropriate columns to fields property
    fields: { value: 'FirstName', groupBy: 'City'},
    //set the placeholder to ComboBox input
    placeholder:"Select an employee",
    //sort the resulted items
    sortOrder: 'Ascending',

```

```
//set the value to groupTemplate
groupTemplate: "<strong>${City}</strong>"
});
//render the component
comboBoxObject.appendTo('#comboelement');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 ComboBox</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:50px auto 0; width:250px;">
    <input type="text" id="comboelement">
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Header template

The header element is shown statically at the top of the popup list items within the ComboBox, and any custom element can be placed as a header element using the [headerTemplate](#) property.

In the following sample, the list items and its headers are designed and displayed as two columns similar to multiple columns of the grid.

INDEX.JS

```
//initiates the component
var comboBoxObject = new ej.dropdowns.ComboBox({
  //bind the data manager instance to dataSource property
  dataSource: new ej.data.DataManager({
```

```

        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/',
        adaptor: new ej.data.ODataV4Adaptor(),
        crossDomain: true
    )),
    //bind the Query instance to query property
    query: new ej.data.Query().from('Employees').select(['FirstName',
'City', 'EmployeeID']).take(6),
    //map the appropriate columns to fields property
    fields: { value: 'FirstName'},
    //set the placeholder to ComboBox input
    placeholder: "Select an employee",
    //sort the resulted items
    sortOrder: 'Ascending',
    //set the value to headerTemplate
    headerTemplate: "<span class='head'><span class='name'>Name</span><span
class='city'>City</span></span>",
    //set the value to item template
    itemTemplate: "<span class='item' ><span
class='name'>${FirstName}</span><span class='city'>${City}</span></span>"
    });
    //render the component
    comboBoxObject.appendTo('#comboelement');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 ComboBox</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin:50px auto 0; width:250px;">
        <input type="text" id="comboelement">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>

```



```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Footer template

The ComboBox has options to show a footer element at the bottom of the list items in the popup list. Here, you can place any custom element as a footer element using the [footerTemplate](#) property.

In the following sample, footer element displays the total number of list items present in the ComboBox.

INDEX.JS

```
var sportsData = [ "Basketball", "Cricket", "Football", "Golf"];
//initiates the component
var comboBoxObject = new ej.dropdowns.ComboBox({
  //bind the data manager instance to dataSource property
  dataSource: sportsData ,
  //set the placeholder to ComboBox input
  placeholder:"Select games",
  //set the value to footer template
  footerTemplate:"<span class='foot'> Total list items: "+
sportsData.length +"</span>"
});

//render the component
comboBoxObject.appendTo('#comboelement');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 ComboBox</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:50px auto 0; width:250px;">

    <input type="text" id="comboelement">
  </div>
<script>
var ele = document.getElementById('container');
```

```
if(ele) {
    ele.style.visibility = "visible";
}

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

No records template

The ComboBox is provided with support to custom design the popup list content when no data is found and no matches found on search with the help of [noRecordsTemplate](#) property.

In the following sample, popup list content displays the notification of no data available.

INDEX.JS

```
//initiates the component
var comboBoxObject = new ej.dropdowns.ComboBox({
    //bind the data manager instance to dataSource property
    dataSource: [],
    //set the placeholder to ComboBox input
    placeholder:"Select an item",
    //set the value to noRecords template
    noRecordsTemplate:"<span class='norecord'> NO DATA AVAILABLE</span>"
});
//render the component
comboBoxObject.appendTo('#comboelement');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 ComboBox</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin:50px auto 0; width:250px;">
        <input type="text" tabindex="1" id="comboelement">
    </div>
<script>
var ele = document.getElementById('container');
```

```
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Action failure template

There is also an option to custom design the popup list content when the data fetch request fails at the remote server. This can be achieved using the [actionFailureTemplate](#) property.

In the following sample, when the data fetch request fails, the ComboBox displays the notification.

INDEX.JS

```
//initiates the component
var comboBoxObject = new ej.dropdowns.ComboBox({
    //bind the data manager instance to dataSource property
    dataSource: new ej.data.DataManager({
        // Here, use the wrong url to display the action failure
        template
            url: 'https://services.odata.org/V4/Northwind/Northwind.svcs/',
            adaptor: new ej.data.ODataV4Adaptor(),
            crossDomain: true
    }),
    //bind the Query instance to query property
    query: new
ej.data.Query().from('Employees').select(['FirstName']).take(6),
    //map the appropriate columns to fields property
    fields: { text: 'FirstName', value: 'EmployeeID' },
    //set the placeholder to ComboBox input
    placeholder: "Select an employee",
    //set the value to action failure template
    actionFailureTemplate: "<span class='action-failure'> Data fetch get
fails</span>"
});
comboBoxObject.appendTo('#comboelement');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 ComboBox</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
```

```
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin:50px auto 0; width:250px;">
        <input type="text" tabindex="1" id="comboelement">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

See Also

- [How to achieve filtering](#)
- [How to group the data using header](#)
- [How to show the list items with icon](#)

Grouping in ##Platform_Name## Combo box control

The ComboBox supports wrapping nested elements into a group based on different categories. The category of each list item can be mapped through the [groupBy](#) field in the data table. The group header is displayed both as inline and fixed headers. The fixed group header content is updated dynamically on scrolling the popup list with its category value.

In the following sample, vegetables are grouped according on its category using `groupBy` field.

INDEX.JS

```
var vegetableData = [
    { Vegetable: 'Cabbage', Category: 'Leafy and Salad', Id: 'item1' },
    { Vegetable: 'Spinach', Category: 'Leafy and Salad', Id: 'item2' },
    { Vegetable: 'Wheat grass', Category: 'Leafy and Salad', Id: 'item3' },
    { Vegetable: 'Yarrow', Category: 'Leafy and Salad', Id: 'item4' },
    { Vegetable: 'Pumpkins', Category: 'Leafy and Salad', Id: 'item5' },
    { Vegetable: 'Chickpea', Category: 'Beans', Id: 'item6' },
    { Vegetable: 'Green bean', Category: 'Beans', Id: 'item7' },
    { Vegetable: 'Horse gram', Category: 'Beans', Id: 'item8' },
    { Vegetable: 'Garlic', Category: 'Bulb and Stem', Id: 'item9' },
    { Vegetable: 'Nopal', Category: 'Bulb and Stem', Id: 'item10' },
    { Vegetable: 'Onion', Category: 'Bulb and Stem', Id: 'item11' }
];
//initiate the ComboBox
var vegetables = new ej.dropdowns.ComboBox({
    //set the grouped data to dataSource property
    dataSource: vegetableData,
    // map the groupBy field with Category column
    fields: { groupBy: 'Category', value: 'Vegetable' },
```

```
// set the placeholder to the ComboBox input
placeholder: "Find a vegetable"
});
//render the ComboBox component
vegetables.appendTo('#comboelement');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 ComboBox</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:50px auto 0; width:250px;">
    <br>
    <input type="text" id="comboelement">
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

HTML select

The ComboBox also supports grouping of list items under specific groups by initiating the `<select>` element using `optgroup`. The nested items are wrapped based on the `<optgroup>` tag that is presents in the `<select>` element

,

```
<select id="selectElement">
```

```
<optgroup label="Beans">
```

```
<option value="1">Chickpea</option>
```

```
<option value="2">Green bean</option>
<option value="3" selected="selected">Horse gram</option>
</optgroup>
<optgroup label="Leafy and Salad">
<option value="4">Spinach</option>
<option value="5">Cabbage</option>
<option value="6">Wheat grass</option>
<option value="7">Yarrow</option>
</optgroup>
</select>
,
```

INDEX.JS

```
// initialize ComboBox component
let comboBoxObject = new ej.dropdowns.ComboBox({
  placeholder: "Select a vegetable",
  popupHeight: "200px"
});
// render initialized ComboBox
comboBoxObject.appendTo('#selectElement');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 ComboBox</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:50px auto 0; width:250px;">
    <br>
    <select id="selectElement">
      <optgroup label="Beans">
```

```

        <option value="1">Chickpea</option>
        <option value="2">Green bean</option>
        <option value="3" selected="selected">Horse gram</option>
    </optgroup>
    <optgroup label="Leafy and Salad">
        <option value="5">Cabbage</option>
        <option value="4">Spinach</option>
        <option value="6">Wheat grass</option>
        <option value="7">Yarrow</option>
    </optgroup>
</select>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customization

The grouping header is also provided with customization option. This allows custom designing using the [groupTemplate](#) property for both inline and fixed headers.

See Also

- [Group Template support to ComboBox.](#)

Filtering in ##Platform_Name## Combo box control

The ComboBox has built-in support to filter data items when `allowFiltering` is enabled. The filter operation starts as soon as you start typing characters in the component.

To display filtered items in the popup, filter the required data and return it to the ComboBox via [updateData](#) method by using the [filtering](#) event.

The following sample illustrates how to query the data source and pass the data to the ComboBox through the `updateData` method in `filtering` event.

INDEX.JS

```

//initiates the component
var comboBoxObject = new ej.dropdowns.ComboBox({
    //bind the data manager instance to dataSource property
    dataSource: new ej.data.DataManager({
        url:
        'https://services.odata.org/V4/Northwind/Northwind.svc/Customers',
        adaptor: new ej.data.ODATAV4Adaptor(),
        crossDomain: true
    }),
    //bind the Query instance to query property
    query: new ej.data.Query().select(['ContactName']),
    //map the appropriate columns to fields property
    fields: { value: 'ContactName' },
    //set the placeholder to ComboBox input

```

```
placeholder: "Find a customer",
//set the filterType to searching operation
filterType: 'StartsWith',
//sort the resulted items
sortOrder: 'Ascending'
});
//render the component
comboBoxObject.appendTo('#comboelement');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 ComboBox</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:50px auto 0; width:250px;">
    <br>
    <input type="text" id="comboelement">
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Limit the minimum filter character

When filtering the list items, you can set the limit for character count to raise remote request and fetch filtered data on the ComboBox. This can be done by manual validation within the filter event handler.

In the following example, the remote request does not fetch the search data until the search key contains three characters.

INDEX.JS


```
//initiates the component
var searchData = new ej.data.DataManager({
    url:
    'https://services.odata.org/V4/Northwind/Northwind.svc/Customers',
    adaptor: new ej.data.ODataV4Adaptor(),
    crossDomain: true
});
var comboBoxObject = new ej.dropdowns.ComboBox({
    dataSource: searchData,
    query: new ej.data.Query().select(['ContactName',
    'CustomerID']).take(7),
    // map the appropriate column
    fields: { text: 'ContactName', value: 'CustomerID' },
    // set placeholder to ComboBox input element
    placeholder: "Select customers",
    //sort the resulted items
    sortOrder: 'Ascending',
    // set true to allowFiltering for enable filtering supports
    allowFiltering: true,
    //bind the filtering event handler
    filtering: (e) => {
        // load overall data when search key empty.
        if(e.text == '') e.updateData(searchData);
        else{
            // restrict the remote request until search key contains 3
            characters.
            if (e.text.length < 3) { return; }
            var query = new ej.data.Query().select(['ContactName',
            'CustomerID']);
            query = (e.text !== '') ? query.where('ContactName', 'startswith',
            e.text, true) : query;
            e.updateData(searchData, query);
        }
    }
});
//render the component
comboBoxObject.appendTo('#comboelement');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 ComboBox</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
```

```
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin:50px auto 0; width:250px;">
        <br>
        <input type="text" id="comboelement">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Change the filter type

While filtering, you can change the filter type to `contains`, `startsWith`, or `endsWith` for string type within the filter event handler.

In the following examples, data filtering is done with `endsWith` type.

INDEX.JS

```
//initiates the component
var comboBoxObject = new ej.dropdowns.ComboBox({
    //bind the data manager instance to dataSource property
    dataSource: new ej.data.DataManager({
        url:
        'https://services.odata.org/V4/Northwind/Northwind.svc/Customers',
        adaptor: new ej.data.ODataV4Adaptor(),
        crossDomain: true
    }),
    query: new ej.data.Query().select(['ContactName',
    'CustomerID']).take(7),
    // map the appropriate column
    fields: { text: 'ContactName', value: 'CustomerID' },
    // set placeholder to ComboBox input element
    placeholder: "Select customers",
    //sort the resulted items
    sortOrder: 'Ascending',
    // set true to allowFiltering for enable filtering supports
    allowFiltering: true,
    //bind the filtering event handler
    filtering: (e) => {
        // load overall data when search key empty.
        if(e.text == '') e.updateData(searchData);
        else{
            var query = new ej.data.Query().select(['ContactName',
            'CustomerID']);
            // change the type of filtering to ends with filtering
```

```

        query = (e.text !== '') ? query.where('ContactName', 'endswith',
e.text, true) : query;
        e.updateData(searchData, query);
    }
}
});
//render the component
comboBoxObject.appendTo('#comboelement');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 ComboBox</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:50px auto 0; width:250px;">
    <br>
    <input type="text" id="comboelement">
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Case sensitive filtering

Data items can be filtered either with or without case sensitivity using the DataManager. This can be done by passing the fourth optional parameter of the `where` clause.

The following example shows how to perform case-sensitive filter.

INDEX.JS

```

//initiates the component
var searchData = new ej.data.DataManager({

```

```

        url:
        'https://services.odata.org/V4/Northwind/Northwind.svc/Customers',
        adaptor: new ej.data.ODataV4Adaptor(),
        crossDomain: true
    });
    var comboBoxObject = new ej.dropdowns.ComboBox({
        dataSource: searchData,
        query: new ej.data.Query().select(['ContactName',
        'CustomerID']).take(7),
        // map the appropriate column
        fields: { text: 'ContactName', value: 'CustomerID' },
        // set placeholder to ComboBox input element
        placeholder: "Select customers",
        //sort the resulted items
        sortOrder: 'Ascending',
        // set true to allowFiltering for enable filtering supports
        allowFiltering: true,
        //bind the filtering event handler
        filtering: (e) => {
            // load overall data when search key empty.
            if(e.text == '') e.updateData(searchData);
            else{
                var query = new ej.data.Query().select(['ContactName',
                'CustomerID']);
                query = (e.text !== '') ? query.where('ContactName', 'startswith',
                e.text, true) : query;
                e.updateData(searchData, query);
            }
        }
    });
    //render the component
    comboBoxObject.appendTo('#comboelement');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 ComboBox</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
    base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
    inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
    dropdowns/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

```

```
<div id="container" style="margin:50px auto 0; width:250px;">
  <br>
  <input type="text" id="comboelement">
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Diacritics Filtering

The ComboBox supports diacritics filtering which will ignore the [diacritics](#) and makes it easier to filter the results in international characters lists when the [ignoreAccent](#) is enabled.

In the following sample, data with diacritics are bound as dataSource for ComboBox.

INDEX.JS

```
// create local data
var data = [
  'Aeróbics',
  'Aeróbics en Agua',
  'Aerografía',
  'Aerodelaje',
  'Águilas',
  'Ajedrez',
  'Ala Delta',
  'Álbumes de Música',
  'Alusivos',
  'Análisis de Escritura a Mano'];
// initialize ComboBox component
var comboObj = new ej.dropdowns.ComboBox({
  //set the local data to dataSource property
  dataSource: data,
  // set the placeholder to ComboBox input element
  placeholder: 'e.g: aero',
  // enabled the ignoreAccent property for ignore the diacritics
  ignoreAccent: true,
  // set true for enable the filtering support.
  allowFiltering: true
});
comboObj.appendTo('#comboelement');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 ComboBox</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
```

```
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin:50px auto 0; width:250px;">
        <br>
        <input type="text" id="comboelement">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

See Also

- [How to achieve autofill while filtering](#)
- [How to group the data using header](#)

Virtualization in ComboBox Component

ComboBox virtualization is a technique used to efficiently render extensive lists of items while minimizing the impact on performance. This method is particularly advantageous when dealing with large datasets because it ensures that only a fixed number of DOM (Document Object Model) elements are created. When scrolling through the list, existing DOM elements are reused to display relevant data instead of generating new elements for each item. This recycling process is managed internally.

During virtual scrolling, the data retrieved from the data source depends on the popup height and the calculation of the list item height. Enabling the [enableVirtualization](#) option in a ComboBox activates this virtualization technique.

When fetching data from the data source, the [actionBegin](#) event is triggered before data retrieval begins. Then, the [actionComplete](#) event is triggered once the data is successfully fetched.

Furthermore, Incremental Search is supported with virtualization in the Combobox component. When a key is typed, the focus is moved to the respective element in the open popup state. In the closed popup state, the popup opens, and focus is moved to the respective element in the popup list based on the typed key. The Incremental Search functionality is well-suited for scenarios involving remote data binding.

When the `enableVirtualization` property is enabled, the `skip` and `take` properties provided by the user within the Query class at the initial state or during the `actionBegin` or `actionComplete` events will not be considered, since it is internally managed and calculated based on certain dimensions with respect to the popup height.

Binding local data

The Combobox can generate its list items through an array of complex data. For this, the appropriate columns should be mapped to the `fields` property. When using virtual scrolling, the list updates based on the scroll offset value, triggering a request to fetch more data from the server. As the data is being fetched, the `actionBegin` event occurs before the data retrieval starts. Once the data retrieval is successful, the `actionComplete` event is triggered, indicating that the data fetch process is complete.

In the following example, `id` column and `text` column from complex data have been mapped to the `value` field and `text` field, respectively.

INDEX.JS

```
var records = [];
for (var i = 1; i <= 150; i++) {
    var item = {
        id: 'id' + i,
        text: "Item " + i,
    };
    records.push(item);
}
//initiates the component
var comboBoxObject = new ej.dropdowns.ComboBox({
    //bind the data source property
    dataSource: records,
    //map the appropriate columns to fields property
    fields: { value: 'id', text: 'text' },
    //set the placeholder to DropDownList input
    placeholder: "Select an Item ",
    //set enableVirtualization property to true
    enableVirtualization: true,
    //set allowFiltering property to true
    allowFiltering: false,
    //set the height of the popup element
    popupHeight: '200px'
});
//render the component
comboBoxObject.appendTo('#comboelement');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 ComboBox</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
```

```
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin:50px auto 0; width:250px;">
        <input type="text" id="comboelement">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Binding Remote data

The Combobox supports retrieval of data from remote data services with the help of **DataManager** component. When using remote data, it initially fetches all the data from the server, triggering the **actionBegin** and **actionComplete** events, and then stores the data locally. During virtual scrolling, additional data is retrieved from the locally stored data, triggering the **actionBegin** and **actionComplete** events at that time as well.

The following sample displays the OrderId from the **Orders** Data Service.

INDEX.JS

```
//initiates the component
var comboBoxObject = new ej.dropdowns.ComboBox({
    //bind the dataSource property
    dataSource: new ej.data.DataManager({
        url: 'https://ej2services.syncfusion.com/js/development/api/orders',
        adaptor: new ej.data.WebApiAdaptor(),
        crossDomain: true
    }),
    //map the appropriate columns to fields property
    fields: { text: 'OrderID', value: 'OrderID' },
    //set the placeholder to DropDownList input
    placeholder: "Select an Item ",
    //set enableVirtualization property to true
    enableVirtualization: true,
    //set allowFiltering property to true
    allowFiltering: true,
    //set the height of the popup element
```



```
popupHeight: '200px'
});
//render the component
comboBoxObject.appendTo('#comboelement');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 ComboBox</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:50px auto 0; width:250px;">
    <input type="text" id="comboelement">
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Grouping with Virtualization

The Combobox component supports grouping with Virtualization. It allows you to organize elements into groups based on different categories. Each item in the list can be classified using the [groupBy](#) field in the data table. After grouping, virtualization works similarly to local data binding, providing a seamless user experience. When the data source is bound to remote data, an initial request is made to retrieve all data for the purpose of grouping. Subsequently, the grouped data works in the same way as local data binding virtualization, enhancing performance and responsiveness.

The following sample shows the example for Grouping with Virtualization.

INDEX.JS

```

var records = [];
for (var i = 1; i <= 150; i++) {
    var dropdownsItem = {};
    dropdownsItem.id = 'id' + i;
    dropdownsItem.text = "Item ".concat(i);
    var randomAutoGroup = Math.floor(Math.random() * 4) + 1;
    switch (randomAutoGroup) {
        case 1:
            dropdownsItem.group = 'Group D';
            break;
        case 2:
            dropdownsItem.group = 'Group C';
            break;
        case 3:
            dropdownsItem.group = 'Group B';
            break;
        case 4:
            dropdownsItem.group = 'Group A';
            break;
        default:
            break;
    }
    records.push(dropdownsItem);
}
//initiates the component
var comboBoxObject = new ej.dropdowns.ComboBox({
    //bind the data source property
    dataSource: records,
    //map the appropriate columns to fields property
    fields: { groupBy: 'group', text: 'text', value: 'id' },
    //set the placeholder to DropDownList input
    placeholder: "Select an Item ",
    //set enableVirtualization property to true
    enableVirtualization: true,
    //set allowFiltering property to true
    allowFiltering: true,
    //set the height of the popup element
    popupHeight: '200px'
});
//render the component
comboBoxObject.appendTo('#comboelement');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 ComboBox</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

```

```
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin:50px auto 0; width:250px;">
        <input type="text" id="comboelement">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Filtering with Virtualization

The ComboBox component supports Filtering with Virtualization. The ComboBox includes a built-in feature that enables data filtering when the [allowFiltering](#) option is enabled. In the context of Virtual Scrolling, the filtering process operates in response to the typed characters. Specifically, the DropDownList sends a request to the server, utilizing the full data source, to achieve filtering. Before initiating the request, an action event is triggered. Upon successful retrieval of data from the server, an action complete event is triggered. The initial data is loaded when the popup is opened. Whether the filter list has a selection or not, the popup closes.

The following sample shows the example for Filtering with Virtualization.

INDEX.JS

```
var records = [];
for (var i = 1; i <= 150; i++) {
    var item = {
        id: 'id' + i,
        text: "Item " + i,
    };
    records.push(item);
}
//initiates the component
var comboBoxObject = new ej.dropdowns.ComboBox({
    //bind the dataSorce property
    dataSource: records,
    //map the appropriate columns to fields property
    fields: { value: 'id', text: 'text' },
    //set the placeholder to DropDownList input
    placeholder: "Select an Item ",
    //set enableVirtualization property to true
```

```
enableVirtualization: true,
//set allowFiltering property to true
allowFiltering: true,
//set the height of the popup element
popupHeight: '200px'
});
//render the component
comboBoxObject.appendTo('#comboelement');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 ComboBox</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:50px auto 0; width:250px;">
    <input type="text" id="comboelement">
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Localization in ##Platform_Name## Combo box control

The Localization library allows you to localize static text content of the [noRecordsTemplate](#) and [actionFailureTemplate](#) properties according to the culture currently assigned to the ComboBox.

| Locale key | en-US (default) |

|-----|-----|

| noRecordsTemplate | No records found |

| actionFailureTemplate | The request failed |

Loading translations

To load translation object to your application, use `load` function of `L10n` class.

In the following sample, French culture is set to the ComboBox and no data is loaded. Hence, the [noRecordsTemplate](#) property displays its text in French culture initially, and if the sample is run offline, the [actionFailureTemplate](#) property displays its text appropriately.

INDEX.JS

```
//bind the data manager instance to dataSource property
var customerData = new ej.data.DataManager({
  url: 'https://services.odata.org/V4/Northwind/Northwind.svc/Customers',
  adaptor: new ej.data.ODataV4Adaptor(),
  crossDomain: true
});
//initiates the component
var comboBoxObject = new ej.dropdowns.ComboBox({
  //set the data to dataSource property
  dataSource: customerData,
  // set locale culture to ComboBox
  locale: 'fr-BE',
  // map appropriate column
  fields: { text: 'ContactName', value: 'CustomerID' },
  // take 0 item to showcase noRecordsTemplate property.
  query: new ej.data.Query().select(['ContactName',
  'CustomerID']).take(0),
  // set placeholder to ComboBox input element
  placeholder: 'Sélectionnez un éléments'
});
comboBoxObject.appendTo('#comboelement');
ej.base.L10n.load({
  'fr-BE': {
    'dropdowns': {
      'noRecordsTemplate': 'Aucun enregistrement trouvé',
      'actionFailureTemplate': 'Modèle d'échec d'action'
    }
  }
});
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 ComboBox</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
```

```
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin:50px auto 0; width:250px;">
        <br>
        <input type="text" id="comboelement">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

See Also

- [Accessibility](#)
- [How to bind the data to the combobox](#)

Style in ##Platform_Name## Combo box control

The following content provides the exact CSS structure that can be used to modify the control's appearance based on the user preference.

Customizing the appearance of wrapper element

Use the following CSS to customize the appearance of wrapper element.

,

```
.e-ddl.e-input-group.e-control-wrapper .e-input {
font-size: 20px;
font-family: emoji;
color: #ab3243;
background: #32a5ab;
}
,
```

Customizing the dropdown icon's color

Use the following CSS to customize the dropdown icon's color.

,

```
.e-ddl.e-input-group .e-input-group-icon,.e-ddl.e-input-group.e-control-wrapper .e-input-group-
icon: hover {
```

```
color: #bb233d;
```

```
font-size: 13px;
```

```
}
```

```
,
```

Customizing the focus color

Use the following CSS to customize the focusing color of input element.

```
,
```

```
.e-ddl.e-input-group.e-control-wrapper.e-input-focus::before, .e-ddl.e-input-group.e-control-wrapper.e-
input-focus::after {
```

```
background: #c000ff;
```

```
}
```

```
,
```

Customizing the outline theme's focus color

Use the following CSS to customize the focusing color of outline theme.

```
,
```

```
.e-outline.e-input-group.e-input-focus: hover: not(.e-success): not(.e-warning): not(.e-error): not(.e-
disabled): not(.e-float-icon-left),.e-outline.e-input-group.e-input-focus.e-control-wrapper: hover: not(.e-
success): not(.e-warning): not(.e-error): not(.e-disabled): not(.e-float-icon-left),.e-outline.e-input-group.e-
input-focus: not(.e-success): not(.e-warning): not(.e-error): not(.e-disabled),.e-outline.e-input-group.e-
control-wrapper.e-input-focus: not(.e-success): not(.e-warning): not(.e-error): not(.e-disabled) {
```

```
border-color: #b1bd15;
```

```
box-shadow: inset 1px 1px #b1bd15, inset -1px 0 #b1bd15, inset 0 -1px #b1bd15;
```

```
}
```

```
,
```

Customizing the disabled component's text color

Use the following CSS to customize the text color when the component is disabled.

```
,
```

```
.e-input-group.e-control-wrapper .e-input[disabled] {
```

```
-webkit-text-fill-color: #0d9133;
```

```
}
```

```
,
```

Customizing the float label element's focusing color

Use the following CSS to customize the focusing color of float label element.

```
,
```

```
.e-float-input.e-input-group:not(.e-float-icon-left) .e-float-line::before,.e-float-input.e-control-
wrapper.e-input-group:not(.e-float-icon-left) .e-float-line::before,.e-float-input.e-input-group:not(.e-
float-icon-left) .e-float-line::after,.e-float-input.e-control-wrapper.e-input-group:not(.e-float-icon-left)
.e-float-line::after {
```

```
background-color: #2319b8;
```

```
}
```

```
.e-ddl.e-input-group.e-control-wrapper.e-float-input.e-input-focus .e-float-text.e-label-top, .e-float-
input.e-control-wrapper:not(.e-error).e-input-focus input ~ label.e-float-text {
```

```
color: #2319b8;
```

```
}
```

```
,
```

Customizing the color of the placeholder text

Use the following CSS to customize the text color of placeholder.

```
,
```

```
.e-ddl.e-input-group input.e-input::placeholder {
```

```
color: red;
```

```
}
```

```
,
```

Customizing the text selection color

Use the following CSS to customize the selection color of text and background.

```
,
```

```
.e-ddl.e-input-group input.e-input::selection {
```

```
color: red;
```

```
background: yellow;
```

```
}
```

```
,
```

Customizing the background color of focus, hover, and active item's

Use the following CSS to customize the background color of focus, hover and active item's.

```
,
```

```
.e-dropdownbase .e-list-item.e-item-focus, .e-dropdownbase .e-list-item.e-active, .e-dropdownbase .e-
list-item.e-active.e-hover, .e-dropdownbase .e-list-item.e-hover {
```

```
background-color: #1f9c99;
```

```
color: #2319b8;
```

```
}
```

```
,
```


Customizing the appearance of pop-up element

Use the following CSS to customize the appearance of popup element.

,

```
.e-dropdownbase .e-list-item, .e-dropdownbase .e-list-item.e-item-focus {
background-color: #29c2b8;
color: #207cd9;
font-family: emoji;
min-height: 29px;
}
```

,

Adding mandatory asterisk to placeholder and float label

You can add a mandatory asterisk(*) to placeholder and float label using `.e-input-group.e-control-wrapper.e-float-input .e-float-text::after` class.

INDEX.JS

```
var sportsData = ['Badminton', 'Cricket', 'Football', 'Golf', 'Tennis'];
// initialize ComboBox component
var listObj = new ej.dropdowns.ComboBox({
  dataSource: sportsData,
  // set placeholder to ComboBox input element
  placeholder: "Select games"});
listObj.appendTo('#comboelement');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 ComboBox</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="asterisk.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>
```

```
<div id="container" style="margin:50px auto 0; width:250px;">
  <br>
  <input type="text" tabindex="1" id="comboelement">
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Accessibility in ##Platform_Name## Combo box control

The ComboBox component has been designed, keeping in mind the WAI-ARIA specifications, and applies the WAI-ARIA roles, states, and properties along with keyboard support. This component is characterized by complete keyboard interaction support and ARIA accessibility support that makes it easy for people who use assistive technologies (AT) or those who completely rely on keyboard navigation.

The ComboBox component followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the ComboBox component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2 Support](#) | |

| [Section 508 Support](#) | |

| [Screen Reader Support](#) | |

| [Right-To-Left Support](#) | |

| [Color Contrast](#) | |

| [Mobile Device Support](#) | |

| [Keyboard Navigation Support](#) | |

| [Accessibility Checker Validation](#) | |

| [Axe-core Accessibility Validation](#) | |

```
<style>
.post .post-content img {
display: inline-block;
margin: 0.5em 0;
}
</style>

<div> - All
features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The
component does not meet the requirement.</div>
```

WAI-ARIA attributes

The ComboBox component uses the **combobox** role, and each list item has an **option** role. The following **ARIA attributes** denote the ComboBox state.

| Properties | Functionalities |

| --- | --- |

| aria-haspopup | Indicates whether the ComboBox input element has a popup list or not. |

| aria-expanded | Indicates whether the popup list has expanded or not. |

| aria-selected | Indicates the selected option. |

| aria-readonly | Indicates the readonly state of the ComboBox element. |

| aria-disabled | Indicates whether the ComboBox component is in a disabled state or not. |

| aria-activedescendent | This attribute holds the ID of the active list item to focus its descendant child element. |

| aria-owns | This attribute contains the ID of the popup list to indicate popup as a child element. |

| aria-autocomplete | This attribute contains the 'both' to a list of options shows and the currently selected suggestion also shows inline. |

Keyboard interaction

You can use the following key shortcuts to access the ComboBox without interruptions.

| Keyboard shortcuts | Actions |

| --- | --- |

| Arrow Down | Selects the first item in the ComboBox when no item selected. Otherwise, selects the item next to the currently selected item. |

| Arrow Up | Selects the item previous to the currently selected one. |

| Page Down | Scrolls down to the next page and selects the first item when popup list opens. |

| Page Up | Scrolls up to the previous page and selects the first item when popup list opens. |

| Enter | Selects the focused item and popup list closes when it is in open state. |

| Tab | Focuses on the next TabIndex element on the page when the popup is closed.
Otherwise, closes the popup list and remains the focus of the component. |

| Shift + tab | Focuses on the previous TabIndex element on the page when the popup is closed. Otherwise, closes the popup list and remains the focus of the component. |

| Alt + Down | Open the popup list |

| Alt + Up | Close the popup list |

| Esc(Escape) | Closes the popup list when it is in an open state and the currently selected item remains the same. |

| Home | Cursor moves to before of first character in input |

| End | Cursor moves to next of last character in input |

In the following sample, focus the ComboBox component using alt+t keys.

INDEX.JS

```
var gameList = [
    { id: 'Game1', game: 'Badminton' },
    { id: 'Game2', game: 'Basketball' },
    { id: 'Game3', game: 'Cricket' },
    { id: 'Game4', game: 'Football' },
    { id: 'Game5', game: 'Golf' },
    { id: 'Game6', game: 'Hockey' },
    { id: 'Game7', game: 'Rugby' },
    { id: 'Game8', game: 'Snooker' },
    { id: 'Game9', game: 'Tennis' },
];
// initialize ComboBox component
var comboBoxObject = new ej.dropdowns.ComboBox({
    //set the data to dataSource property
    dataSource: gameList,
    //map to column to fields
    fields: { text: 'game', value: 'id' },
    // set placeholder to ComboBox input element
    placeholder: "Select games",
    // set the popup list height
    popupHeight: '200px'
});
// render initialized ComboBox
comboBoxObject.appendTo('#comboelement');
document.onkeyup = function (e) {
    if (e.altKey && e.keyCode === 84 /* t */) {
        // press alt+t to focus the control.
        comboBoxObject.focusIn();
    }
};
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 ComboBox</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:50px auto 0; width:250px;">
    <br>
    <input type="text" tabindex="1" id="comboelement">
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Ensuring accessibility

The ComboBox component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the ComboBox component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the ComboBox component with accessibility tools.

See also

- [Accessibility in Syncfusion components](#)

How To

Autofill in ##Platform_Name## Combo box control

The ComboBox supports the **autofill** behaviour with the help of [autofill](#) property. Whenever you change the input value, the ComboBox will autocomplete your data by matching the typed character. Suppose, if no matches found then, comboBox doesn't suggest any item.

In the following sample, showcase that how to work autofill with ComboBox.

INDEX.JS

```
let searchData = [
  { Name: 'Australia', Code: 'AU' },
  { Name: 'Bermuda', Code: 'BM' },
  { Name: 'Canada', Code: 'CA' },
  { Name: 'Cameroon', Code: 'CM' },
  { Name: 'Denmark', Code: 'DK' },
  { Name: 'France', Code: 'FR' },
  { Name: 'Finland', Code: 'FI' },
  { Name: 'Germany', Code: 'DE' },
  { Name: 'Greenland', Code: 'GL' },
  { Name: 'Hong Kong', Code: 'HK' },
  { Name: 'India', Code: 'IN' },
  { Name: 'Italy', Code: 'IT' },
  { Name: 'Japan', Code: 'JP' },
  { Name: 'Mexico', Code: 'MX' },
  { Name: 'Norway', Code: 'NO' },
  { Name: 'Poland', Code: 'PL' },
  { Name: 'Switzerland', Code: 'CH' },
  { Name: 'United Kingdom', Code: 'GB' },
  { Name: 'United States', Code: 'US' }
];
//initiates the component
let comboBoxObject = new ej.dropdowns.ComboBox({
  // bind the country data to dataSource property
  dataSource: searchData,
  // maps the appropriate column to fields property
  fields: { value: "Name" },
  //set the placeholder to ComboBox input
  placeholder: "Find a country",
  //enable the autofill property to fill a first matched value in input
  //when press a down key
  autofill: true
});
comboBoxObject.appendTo('#comboelement');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 ComboBox</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
```

```
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin:50px auto 0; width:250px;">
        <br>
        <input type="text" tabindex="1" id="comboelement">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Cascading in ##Platform_Name## Combo box control

The cascading ComboBox is a series of ComboBox, where the value of one ComboBox depends upon another's value. This can be configured by using the [change](#) event of the parent ComboBox. Within that change event handler, data has to be loaded to the child ComboBox based on the selected value of the parent ComboBox.

The following example, shows the cascade behavior of country, state, and city ComboBox. Here, the [dataBind](#) method is used to reflect the property changes immediately to the ComboBox.

INDEX.JS

```
//define the country ComboBox data
var countryData = [
    { CountryName: 'Australia', CountryId: '2' },
    { CountryName: 'United States', CountryId: '1' }
];
//define the state ComboBox data
var stateData = [
    { StateName: 'New York', CountryId: '1', StateId: '101' },
    { StateName: 'Virginia ', CountryId: '1', StateId: '102' },
    { StateName: 'Tasmania ', CountryId: '2', StateId: '105' }
];
//define the city ComboBox data
var cityData = [
    { CityName: 'Albany', StateId: '101', CityId: 201 },
    { CityName: 'Beacon ', StateId: '101', CityId: 202 },
    { CityName: 'Emporia', StateId: '102', CityId: 206 },
    { CityName: 'Hampton ', StateId: '102', CityId: 205 },
    { CityName: 'Hobart', StateId: '105', CityId: 213 },
    { CityName: 'Launceston ', StateId: '105', CityId: 214 }
];
//initiates the country ComboBox
var countryObj = new ej.dropdowns.ComboBox({
    //set the data to dataSource property
    dataSource: countryData,
    fields: { value: 'CountryId', text: 'CountryName' },
```

```

//bind the change event handler
change: () => {
    //Query the data source based on country ComboBox selected value
    stateObj.query = new ej.data.Query().where('CountryId', 'equal',
countryObj.value);
    // enable the state ComboBox
    stateObj.enabled = true;
    //clear the existing selection.
    stateObj.text = null;
    // bind the property changes to state ComboBox
    stateObj.dataBind();
    //clear the existing selection in city ComboBox
    cityObj.text = null;
    //disabe the city ComboBox
    cityObj.enabled = false;
    //bind the property cahnges to City ComboBox
    cityObj.dataBind();
},
placeholder: 'Select a country',
});
//render the country ComboBox
countryObj.appendTo('#countries');
//initiates the state ComboBox
var stateObj = new ej.dropdowns.ComboBox({
    //set the data to dataSource property
    dataSource: stateData,
    fields: { value: 'StateId', text: 'StateName' },
    // set disable state by default to prevent user interact.
    enabled: false,
    change: () => {
        // Query the data source based on state ComboBox selected value
        cityObj.query = new ej.data.Query().where('StateId', 'equal',
stateObj.value);
        // enable the city ComboBox
        cityObj.enabled = true;
        //clear the existing selection
        cityObj.text = null;
        // bind the property change to city ComboBox
        cityObj.dataBind();
    },
    placeholder: 'Select a state',
});
//render the state ComboBox
stateObj.appendTo('#states');
//initiates the city ComboBox
var cityObj = new ej.dropdowns.ComboBox({
    //set the data to dataSource property
    dataSource: cityData,
    fields: { text: 'CityName', value: 'CityId' },
    // disable the ComboBox by default to prevent the user interact.
    enabled: false,
    placeholder: 'Select a city',
});
//render the city ComboBox
cityObj.appendTo('#cities');

```


INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 ComboBox</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:50px auto 0; width:250px;">
    <br>
    <input type="text" id="countries">
    <div class="padding-top">
      <input type="text" id="states">
    </div>
    <div class="padding-top">
      <input type="text" id="cities">
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Icons support in ##Platform_Name## Combo box control

You can render **icons** to the list items by mapping the [iconCss](#) field. This `iconCss` field create a span in the list item with mapped class name to allow styling as per your need.

In the following sample, icon classes are mapped with `iconCss` field.

INDEX.JS

```
let sortFormatData = [
  { Class: 'asc-sort', Type: 'Sort A to Z', Id: '1' },
  { Class: 'dsc-sort', Type: 'Sort Z to A ', Id: '2' },
  { Class: 'filter', Type: 'Filter', Id: '3' },
  { Class: 'clear', Type: 'Clear', Id: '4' }
```

```

];
//initiates the component
let sortFormat = new ej.dropdowns.ComboBox({
  //set the data to dataSource property
  dataSource: sortFormatData,
  // map the icon column to iconCSS field.
  fields: { value: 'Type', iconCss: 'Class' },
  // set placeholder to ComboBox input element
  placeholder: 'Find a format'
});
sortFormat.appendTo('#comboelement');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 ComboBox</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:50px auto 0; width:250px;">
    <br>
    <input type="text" id="comboelement">
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Achieve virtual scrolling in ##Platform_Name## Combo box control

The Virtual Scrolling is used to display a large amount of data without buffering the entire load of a huge database record in the ComboBox, that is, when scrolling, the request is sent and fetch some amount of data from the server dynamically. Using the `scroll` event, get the data and generate the list add to popup using the `addItem` method.

Refer to the following code sample for virtual scrolling.

INDEX.JS

```
var data = new ej.data.DataManager({
    url: 'https://js.syncfusion.com/demos/ejServices/Wcf/Northwind.svc/',
    crossDomain: true
});
//initiates the component
let comboObject = new ej.dropdowns.ComboBox({
    // bind the DataManager instance to dataSource property
    dataSource: data,
    // bind the Query instance to query property
    query: new
ej.data.Query().from('Customers').select('ContactName').take(7),
    // map the appropriate columns to fields property
    fields: { text: 'ContactName', value: 'ContactName' },
    // set the placeholder to ComboBox input element
    placeholder: 'Select a customer',
    // sort the resulted items
    sortOrder: 'Ascending',
    // set the height of the popup element
    popupHeight: '200px',

    actionComplete: function (e) {
        let operator = new
ej.data.Query().from('Customers').select('ContactName');
        let start = 7;
        let end = 12;
        let listElement = this.list;
        listElement.addEventListener('scroll', () => {
            if ((listElement.scrollTop + listElement.offsetHeight >=
listElement.scrollHeight)) {
                let filterQuery = operator.clone();
                data.executeQuery(filterQuery.range(start,
end)).then((event) => {
                    start = end;
                    end += 5;
                    comboObject.addItem(event.result);
                }).catch((e) => {
                });
            }
        })
    }
});
comboObject.appendTo('#atcelement');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
<title>Essential JS 2 AutoComplete</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="styles.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin:0 auto; width:250px;">
        <input type="text" tabindex="1" id="atcelement">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

ContextMenu

Icons and navigation in ##Platform_Name## Context menu control

Icons

The ContextMenu item have an icon/image in it to provide visual representation of the action. To place the icon on a menu item, set the [iconCss](#) property to e-icons with the required icon CSS. By default, the icon is positioned to the left side of the menu item. In the following sample, the icons for Cut, Copy and Paste menu items are added using the [iconCss](#) property.

INDEX.JS

```

ej.base.enableRipple(true);
var menuItems = [
    {
        text: 'Cut',
        iconCss: 'e-db-icons e-cut'
    },
    {
        text: 'Copy',
        iconCss: 'e-icons e-copy'
    },
    {
        text: 'Paste',
        iconCss: 'e-db-icons e-paste',
    }
];
var menuOptions = {
    target: '#target',

```

```

        items: menuItems
    };
    // Initialize ContextMenu component.
    var menuObj = new ej.navigations.ContextMenu(menuOptions, '#contextmenu')

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <!--target element-->
    <div id="target">Right click / Touch hold to open the
ContextMenu</div>
    <!--element which is going to render-->
    <ul id="contextmenu"></ul>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Navigation

Navigation in ContextMenu is usage to navigate to the other web page when menu item is clicked. This can be achieved by providing link to the menu item using the [url](#) property. In the following sample, Navigation URL for Flipkart, Amazon, and Snapdeal menu items are added using the [url](#) property.

INDEX.JS

```
ej.base.enableRipple(true);
var menuItems = [
  {
    text: 'Flipkart',
    iconCss: 'e-cart-icon e-link',
    url: 'https://www.google.co.in/search?q=flipkart'
  },
  {
    text: 'Amazon',
    iconCss: 'e-cart-icon e-link',
    url: 'https://www.google.co.in/search?q=amazon'
  },
  {
    text: 'Snapdeal',
    iconCss: 'e-cart-icon e-link',
    url: 'https://www.google.co.in/search?q=snapdeal'
  }
];
var menuOptions = {
  target: '#target',
  items: menuItems,
  beforeItemRender: beforeItemRender
};
var menuObj = new ej.navigations.ContextMenu(menuOptions, '#contextmenu')
function beforeItemRender(args) {
  args.element.getElementsByTagName('a')[0].setAttribute('target',
  '_blank');
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
  user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  navigations/styles/material.css" rel="stylesheet">
```

```

<!--style reference from app-->
<link href="styles.css" rel="stylesheet">
<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <!--target element-->
        <div id="target">Right click / Touch hold to open the
ContextMenu</div>
        <!--element which is going to render-->
        <ul id="contextmenu"></ul>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

To open the links in new tab, set `target` attribute with the value `_blank` in the [beforeItemRender](#) event.

See Also

- [How to change menu items dynamically](#)

Template and multilevel nesting in ##Platform_Name## Context menu control

Template

The ContextMenu items can be customized by using the [beforeItemRender](#) event. The item render event

triggers while rendering each menu item. The event argument will be used to identify the menu item and customize it based on the requirement. In the following sample, the menu item is rendered with keycode for specified action in ContextMenu using the template. Here, the keycode is specified for Save as, View page source, and Inspect in the right side corner of the menu items by adding span element in the [beforeItemRender](#) event.

INDEX.JS

```

ej.base.enableRipple(true);
var menuItems = [
    {
        text: 'Save as...'
    },
    {
        text: 'View page source'
    }
]

```

```

    },
    {
        text: 'Inspect'
    }
    ]];
var menuOptions = {
    target: '#target',
    items: menuItems,
    beforeItemRender: beforeItemRender
};
var menuObj = new ej.navigations.ContextMenu(menuOptions, '#contextmenu');
function beforeItemRender(args) {
    var shortCutSpan = ej.base.createElement('span');
    var text = args.item.text;
    var shortCutText = text === 'Save as...' ? 'Ctrl + S' : (text === 'View
page source' ? 'Ctrl + U' : 'Ctrl + Shift + I');
    shortCutSpan.textContent = shortCutText;
    args.element.appendChild(shortCutSpan);
    shortCutSpan.setAttribute('class', 'shortcut');
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Essential JS 2">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <!--style reference from app-->
    <link href="styles.css" rel="stylesheet">
    <!--system js reference and configuration-->

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <!--target element-->

```



```

        <div id="target">Right click / Touch hold to open the
ContextMenu</div>
        <!--element which is going to render-->
        <ul id="contextmenu"></ul>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

To create span element, `createElement` utility function used from `ej2-base`.

Multilevel nesting

The Multiple level nesting supports in ContextMenu. It can be achieved by mapping the `items` property inside the parent `menuitems`. In the below sample, three level nesting of ContextMenu is provided.

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Essential JS 2">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <!--style reference from app-->
    <link href="styles.css" rel="stylesheet">
    <!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <!--target element-->

```

```

    <div id="target">Right click / Touch hold to open the
ContextMenu</div>
    <!--element which is going to render-->
    <ul id="contextmenu"></ul>
  </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

To open sub menu items only on click, [showItemOnClick](#) property should be set as `true`.

See Also

- [Populate menu items with data source](#)

Accessibility in ##Platform_Name## Context menu control

The Context menu component followed the accessibility guidelines and standards, including [ADA](#), [Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Context menu component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support | |

| [Section 508](#) Support | |

| Screen Reader Support | |

| Right-To-Left Support | |

| Color Contrast | |

| Mobile Device Support | |

| Keyboard Navigation Support | |

| [Accessibility Checker](#) Validation | |

| [Axe-core](#) Accessibility Validation | |

```
<style>
.post .post-content img {
display: inline-block;
margin: 0.5em 0;
}
</style>

<div> - All
features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The
component does not meet the requirement.</div>
```

WAI-ARIA attributes

The Context menu component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Context menu component:

Attributes Purpose
--- ---
role Indicates Context menu component popup as menu , and the popup items as menuitem .
aria-haspopup Indicates the availability and type of interactive popup element.
aria-expanded Indicates whether the subtree can be expanded or collapsed, as well as indicates whether its current state is expanded or collapsed.
aria-label Indicates the menu item text.

Keyboard interaction

The Context menu component followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Context menu component.

Press To do this
--- ---
Esc Closes the opened sub menu.
Enter Selects the focused item.
Up Navigates up or to the previous menu item.
Down Navigates down or to the next menu item.
Left Close the current sub menu and navigates to the parent menu.
Right Navigates and open the next sub menu.

Ensuring accessibility

The Context menu component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Context menu component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Context menu component with accessibility tools.

See also

- [Accessibility in Syncfusion ##Platform_Name## components](#)

Style and appearance in ##Platform_Name## Context menu control

To modify the ContextMenu appearance, you need to override the default CSS of ContextMenu component. Please find the list of CSS classes and its corresponding section in ContextMenu component. Also, you have an option to create your own custom theme for the controls using our [Theme Studio](#).

CSS Class | Purpose of Class

|.e-contextmenu-wrapper |To customize the context menu wrapper

|.e-contextmenu-wrapper .e-menu-parent|To customize the context menu items

|.e-contextmenu-wrapper ul .e-menu-item.e-selected .e-caret::before|To customize the context menu caret icon

|.e-contextmenu-wrapper ul .e-menu-item .e-menu-icon::before|To customize the icons of the context menu

How To

Populate menu items with data source in ##Platform_Name## Context menu control

To bind local data source to the ContextMenu, menu items are populated from data source and mapped to [items](#) property.

The below example demonstrates how to bind local data source to the ContextMenu and separator is added using [insertAfter](#) method.

INDEX.JS

```
ej.base.enableRipple(true);  
var data = [  
  { id: 1, parentId: null, text: 'View' },  
  { id: 2, parentId: null, text: 'Sort by' },  
  { id: 3, parentId: null, text: '' },  
  { id: 4, parentId: null, text: 'New' },  
  { id: 5, parentId: null, text: '' },  
  { id: 6, parentId: null, text: 'Display Settings' },  
  { id: 7, parentId: null, text: 'Personalize' },  
  //first level child  
  { id: 8, parentId: 1, text: 'Large Icons' },  
  { id: 9, parentId: 1, text: 'Medium Icons' },  
  { id: 10, parentId: 1, text: 'Small Icons' },  
  { id: 11, parentId: 2, text: 'Name' },  
  { id: 12, parentId: 2, text: 'Size' },  
  { id: 13, parentId: 4, text: 'Folder' },  
  { id: 14, parentId: 4, text: 'Shortcut' },
```

```

    { id: 15, parentId: 4, text: '' },
    { id: 16, parentId: 4, text: 'Contact' }
  ];
  var record;
  var menuItems = [];
  for (var i = 0; i < data.length; i++) {
    record = data[i];
    if (record.parentId) {
      if (!menuItems[record.parentId - 1].items) {
        menuItems[record.parentId - 1].items = []
      }
      menuItems[record.parentId - 1].items.push({ text: record.text });
    } else {
      menuItems.push({ text: record.text });
    }
  }
  var menuOptions = {
    target: '#target',
    items: menuItems,
    beforeItemRender: (args) => {
      if (!args.item.text) {
        args.element.classList.add('e-separator');
      }
    }
  };
  var menuObj = new ej.navigations.ContextMenu(menuOptions, '#contextmenu');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">

  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>

```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <!--target element-->
        <div id="target">Right click / Touch hold to open the
ContextMenu</div>
        <!--element which is going to render-->
        <ul id="contextmenu"></ul>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Open and close contextmenu in ##Platform_Name## Context menu control

ContextMenu can be opened and closed programmatically whenever required by using the open and close methods.

In the following example, the ContextMenu is opened using the [open](#) method at the specified position using [top](#) and [left](#). Also, ContextMenu is closed using [close](#) method on ContextMenu item click or document click.

INDEX.JS

```

ej.base.enableRipple(true);
var menuItems = [
    {
        text: 'Cut'
    },
    {
        text: 'Copy'
    },
    {
        text: 'Paste'
    }
];
var menuOptions = {
    items: menuItems
};
var menuObj = new ej.navigations.ContextMenu(menuOptions, '#contextmenu');
var button = new ej.buttons.Button();
button.appendTo('#btnElement');
document.getElementById('btnElement').onclick=function() {
    var contextMenuObj =
document.getElementById("contextmenu").ej2_instances[0];
    contextMenuObj.open(60, 20);
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">

  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <!--element which is going to render-->
    <ul id="contextmenu"></ul>
    <button class="e-btn" id="btnElement">Open ContextMenu</button>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Change menu items dynamically in ##Platform_Name## Context menu control

The items visible in the ContextMenu can be changed dynamically based on the target in which you open the ContextMenu. To achieve this behavior, initialize ContextMenu with all items using [items](#) property and then based on the context you open hide/show required items using [hideItems](#)/[showItems](#) method in [beforeOpen](#) event.

In the following example, the datasource for Clipboard div is **Cut**, **Copy**, **Paste** and for the Editor div is **Add**, **Edit**, **Delete** is changed on [beforeOpen](#) event using [hideItems](#) and [showItems](#) method.

INDEX.JS

```

ej.base.enableRipple(true);
var menuItems = [
    {
        text: 'Cut'
    },
    {
        text: 'Copy'
    },
    {
        text: 'Paste'
    },
    {
        text: 'Add'
    },
    {
        text: 'Edit'
    },
    {
        text: 'Delete'
    }
];
var menuOptions = {
    target: '#target .e-div',
    items: menuItems,
    beforeOpen: beforeOpen
};
// Initialize ContextMenu component.
var menuObj = new ej.navigations.ContextMenu(menuOptions, '#contextmenu');
function beforeOpen (args) {
    if (args.event.target.id === 'right') {
        menuObj.hideItems(['Cut', 'Copy', 'Paste']);
        menuObj.showItems(['Add', 'Edit', 'Delete']);
    } else if (args.event.target.id === 'left') {
        menuObj.showItems(['Cut', 'Copy', 'Paste']);
        menuObj.hideItems(['Add', 'Edit', 'Delete']);
    }
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Essential JS 2">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<!--style reference from app-->
<link href="styles.css" rel="stylesheet">
<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="target">
            <div id="left" class="e-div">Clipboard</div>
            <div id="right" class="e-div">Editor</div>
        </div>
        <!--element which is going to render-->
        <ul id="contextmenu"></ul>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Template in ##Platform_Name## Context menu control

Render UL and LI template

Add the HTML UL tag with **id** attribute as **#contextmenu** in your **index.html** file with required LI tags and also add target element on which the ContextMenu has to be opened.

INDEX.JS

```

ej.base.enableRipple(true);
var menuObj = new ej.navigations.ContextMenu({ target: '#target' },
'#contextmenu');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
<title>Essential JS 2</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
<meta name="description" content="Essential JS 2">
<meta name="author" content="Syncfusion">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<!--style reference from app-->
<link href="styles.css" rel="stylesheet">
<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <!--target element-->
        <div id="target">Right click / Touch hold to open the
ContextMenu</div>
        <!--element which is going to render-->
        <ul id="contextmenu" class="list">
            <li>Cut</li>
            <li>Copy</li>
            <li>Paste</li>
        </ul>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Show table in sub ContextMenu

Menu items of the ContextMenu can be customized according to the requirement. The section explains about how to customize table template

in sub menu item.

This can be achieved by appending table layout while `li` rendering by using [beforeItemRender](#) event.

INDEX.JS

```

ej.base.enableRipple(true);
var header = document.createElement('h4');
header.textContent = 'Insert Table';
var table = document.createElement('table');

```

```

for (var i = 0; i < 5; i++) {
    var row = document.createElement('tr');
    table.appendChild(row);
    for (var j = 0; j < 6; j++) {
        var col = document.createElement('td');
        row.appendChild(col);
        col.setAttribute('class', 'e-border');
    }
}
var menuItems = [
    {
        text: 'Cut',
        iconCss: 'e-cm-icons e-cut'
    },
    {
        text: 'Copy',
        iconCss: 'e-icons e-copy'
    },
    {
        text: 'Paste',
        iconCss: 'e-cm-icons e-paste'
    },
    {
        separator: true
    },
    {
        text: 'Link',
        iconCss: 'e-icons e-link'
    },
    {
        text: 'Table',
        iconCss: 'e-icons e-table',
        items: [
            {
                id: 'table'
            }
        ]
    }
];
var menuOptions = {
    target: '#target',
    items: menuItems,
    beforeItemRender: function(args) {
        if (args.item.id === 'table') {
            args.element.classList.add('bg-transparent');
            args.element.appendChild(header);
            args.element.appendChild(table);
        }
    }
};
var menuObj = new ej.navigations.ContextMenu(menuOptions, '#contextmenu');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2</title>
    <meta charset="utf-8">

```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
<meta name="description" content="Essential JS 2">
<meta name="author" content="Syncfusion">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<!--style reference from app-->
<link href="styles.css" rel="stylesheet">
<!--system js reference and configuration-->

<style>
.e-contextmenu-wrapper ul .e-menu-item.bg-transparent {
    background-color: transparent;
    line-height: normal;
    height: auto;
}
h4 {
    text-align: center;
    margin-top: 5px;
    margin-bottom: 5px;
}
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="target">Right click / Touch hold to open the
ContextMenu</div>
        <ul id="contextmenu"></ul>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Show UI components in ContextMenu

UI components can also be placed inside the each `li` element of ContextMenu.

In the following example, CheckBox component is placed inside each `li` element and this can be achieved by creating CheckBox component in [beforeItemRender](#) event and appending it into the `li` element.

INDEX.JS

```
ej.base.enableRipple(true);
var menuItems = [
    { text: 'Option 1' },
    { text: 'Option 2' },
    { text: 'Option 3' }
];
var i = 1;
var menuOptions = {
    target: '#target',
    items: menuItems,
    beforeItemRender: function(args) {
        var checkbox = new ej.buttons.CheckBox({ label: 'Option'+ i,
checked: i%2 === 0 ? true : false });
        args.element.innerHTML = '';
        checkbox.appendTo('#checkbox'+i);
        var checkbox = document.getElementsByClassName('e-checkbox-
wrapper');
        args.element.appendChild(checkbox[0]);
        i++;
    },
    beforeClose: function(args) {
        args.cancel = true;
    },
    select: function(args) {
        var selectedElem = args.element.parentElement.querySelectorAll('.e-
selected');
        for(var i=0; i < selectedElem.length; i++){
            var ele = selectedElem[i];
            ele.classList.remove('e-selected');
        }
        var checkbox = args.element.childNodes[0];
        var frame = checkbox.querySelector('.e-frame');
        if (checkbox && frame.classList.contains('e-check')) {
            frame.classList.remove('e-check');
        } else if (checkbox) {
            frame.classList.add('e-check');
        }
    }
};
var menuObj = new ej.navigations.ContextMenu(menuOptions, '#contextmenu');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
```

```

<meta name="description" content="Essential JS 2">
<meta name="author" content="Syncfusion">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<!--style reference from app-->
<link href="styles.css" rel="stylesheet">
<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="target">Right click / Touch hold to open the
ContextMenu</div>
        <ul id="contextmenu"></ul>
        <input type="checkbox" id="checkbox1" style="display:none">
        <input type="checkbox" id="checkbox2" style="display:none">
        <input type="checkbox" id="checkbox3" style="display:none">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Underline a character in the item text in ##Platform_Name## Context menu control

Underline a particular character in a text can be handled in [beforeItemRender](#) event by

adding `<u>` tag in between the text and given as innerHTML in `li` rendering.

INDEX.JS

```

ej.base.enableRipple(true);
var menuItems = [
    {
        text: 'Cut'
    },
    {

```

```

        text: 'Copy'
    },
    {
        text: 'Paste'
    }
    ]];
var menuOptions = {
    target: '#target',
    items: menuItems,
    beforeItemRender: function(args) {
        if (args.item.text === 'Copy') {
            args.element.innerHTML = '<u>C</u>opy';
        }
    }
};
// Initialize ContextMenu component.
var menuObj = new ej.navigations.ContextMenu(menuOptions, '#contextmenu');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Essential JS 2">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <!--style reference from app-->
    <link href="styles.css" rel="stylesheet">
    <!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="target">Right click / Touch hold to open the
ContextMenu</div>
        <ul id="contextmenu"></ul>
    </div>
</body>
</html>
```

```

var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Open a dialog on contextmenu item click in **##Platform_Name##** Context menu control

This section explains about how to open a dialog on ContextMenu item click. This can be achieved by handling dialog open in [select](#) event of the ContextMenu.

In the following sample, Dialog will open while clicking **Save As...** item.

INDEX.JS

```

ej.base.enableRipple(true);
var dialog = new ej.popups.Dialog({
    content: "This file can be saved as PDF",
    buttons: [{
        buttonModel: {
            isPrimary: true,
            content: 'Submit',
            cssClass: 'e-flat',
        },
        click: function () {
            this.hide();
        }
    }],
    target: document.getElementById("container"),
    width: '200px',
    height: '110px'
});
// Render initialized Dialog
dialog.appendTo('#dialog');
dialog.hide();
var menuItems = [
    {
        text: 'Back'
    },
    {
        text: 'Forward'
    },
    {
        text: 'Reload'
    },
    {
        separator: true
    },
    {
        text: 'Save As...'
    },
    {
        text: 'Print'
    },
    {

```



```

        text: 'Cast'
    }
    });
    var menuOptions = {
        target: '#target',
        items: menuItems,
        select: function(args) {
            if(args.item.text === 'Save As...') {
                dialog.show();
            }
        }
    };
    var menuObj = new ej.navigations.ContextMenu(menuOptions, '#contextmenu');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Essential JS 2">
    <meta name="author" content="Syncfusion">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <!--style reference from app-->
    <link href="styles.css" rel="stylesheet">
    <!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="target">Right click / Touch hold to open the
ContextMenu</div>
        <ul id="contextmenu"></ul>
        <div id="dialog"></div>
    </div>
</script>

```

```

var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Change animation settings in ##Platform_Name## Context menu control

To change the animation of the ContextMenu, [animationSettings](#) property is used. The supported effects for ContextMenu are,

Effect	Functionality
-----	-----
None	Specifies the sub menu transform with no animation effect.
SlideDown	Specifies the sub menu transform with slide down effect.
ZoomIn	Specifies the sub menu transform with zoom in effect.
FadeIn	Specifies the sub menu transform with fade in effect.

The following sample illustrates how to open ContextMenu with **FadeIn** effect with the **duration** of **800ms**.

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>

```

```

</head>
<body>

    <div id="container">
        <!--target element-->
        <div id="target">Right click / Touch hold to open the
ContextMenu</div>
        <!--element which is going to render-->
        <ul id="contextmenu"></ul>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Add or remove context menu items in ##Platform_Name## Context menu control

ContextMenu items can be added or removed using the [insertAfter](#), [insertBefore](#) and [removeItems](#) methods.

In the following example, the **Display Settings** menu items are added before the **Personalize** item, the **Sort By** menu items are added after the **Refresh**, and the **Paste** item is removed from context menu.

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Essential JS 2">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <!--style reference from app-->
    <link href="styles.css" rel="stylesheet">
    <!--system js reference and configuration-->

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>

```

```

</head>
<body>

    <div id="container">
        <!--target element-->
        <div id="target">Right click / Touch hold to open the
ContextMenu</div>
        <!--element which is going to render-->
        <ul id="contextmenu"></ul>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Enable or disable context menu items in ##Platform_Name## Context menu control

You can enable and disable the menu items using the [enableItems](#) method in ContextMenu. To enable menuitems, set the `enable` property in argument to `true` and vice-versa.

In the following example, the **Display Settings** in parent items and **Medium icons** in sub menu items are disabled.

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Essential JS 2">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <!--style reference from app-->
    <link href="styles.css" rel="stylesheet">
    <!--system js reference and configuration-->

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>

```

```

</head>
<body>

    <div id="container">
        <!--target element-->
        <div id="target">Right click / Touch hold to open the
ContextMenu</div>
        <!--element which is going to render-->
        <ul id="contextmenu"></ul>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

To disable sub menu items, use the [beforeOpen](#) event.

DashboardLayout

Setting size of cells in ##Platform_Name## Dashboard layout control

The entire layout dimensions are assigned based on the height and width of the parent element. Hence a responsive or static layout can be created by assigning a percentage or static dimension values to the parent element. The layout adapts to mobile resolutions by transforming the entire layout into a stacked orientation so that the panels will be displayed in a vertical column.

The **Dashboard Layout** is a grid structured component which can be split into subsections of equal size known as cells. The total number of cells in each row is defined using the [columns](#) property of the component. The width of each cell will be auto calculated based on total number of cells placed in a row and the height of a cell will be same as that of its width. However, the height of these cells can also be configured to any desired size using the [cellAspectRatio](#) property (cellwidth/cellheight ratio) which defines the cell width to height ratio.

The number of rows within the layout has no limits and can have any number of rows based on the panels count and position. Panels which acts as data containers will be placed or positioned over these cells.

Modifying cell size

In a dashboard, the data to be held by the panel in a cell may be of different size, hence different cell dimensions may be required in different scenarios. In this case, the size of these grid cells can be modified to the required size using the [columns](#) and [cellAspectRatio](#) properties.

The following sample demonstrates how to modify a cell size using [columns](#) and [cellAspectRatio](#) properties. In the below sample the width of the parent element is divided into 5 equal cells based on the columns property value resulting the width of each cell as 100px. The height of these cells will be 50px based on the cellAspectRatio value 100/50 (i.e. for every 100px of width, 50px will be the height of the cell).

INDEX.JS

```

var dashboard = new ej.layouts.DashboardLayout({

```

```

    cellSpacing: [10, 10],
    columns: 5,
    cellAspectRatio: 100/50,
    //Dashboard Layout's dragstart event
    dragStart: onDragStart,
    //Dashboard Layout's drag event
    drag: onDrag,
    //Dashboard Layout's dragstop event
    dragStop: onDragStop,
    panels: [{ "sizeX": 1, "sizeY": 1, "row": 0, "col": 0, content: '<div
class="content">0</div>' },
    { "sizeX": 3, "sizeY": 2, "row": 0, "col": 1, content: '<div
class="content">1</div>' },
    { "sizeX": 1, "sizeY": 3, "row": 0, "col": 4, content: '<div
class="content">2</div>' },
    { "sizeX": 1, "sizeY": 1, "row": 1, "col": 0, content: '<div
class="content">3</div>' },
    { "sizeX": 2, "sizeY": 1, "row": 2, "col": 0, content: '<div
class="content">4</div>' },
    { "sizeX": 1, "sizeY": 1, "row": 2, "col": 2, content: '<div
class="content">5</div>' },
    { "sizeX": 1, "sizeY": 1, "row": 2, "col": 3, content: '<div
class="content">6</div>' }
    ]
  });
  dashboard.appendTo('#dashboard_layout');
  //Dashboard Layout's dragstart event function
  function onDragStart(args) {
    console.log("Drag start");
  }
  //Dashboard Layout's drag event function
  function onDrag(args) {
    console.log("Dragging");
  }
  //Dashboard Layout's dragstop event function
  function onDragStop(args) {
    console.log("Drag stop");
  }
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DashboardLayout </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 DashboardLayout
Component">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <div id="container">
        <!--element which is going to render the dashboardlayout-->
        <div id="dashboard_layout"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Setting cell spacing

The spacing between each panel in a row and column can be defined using the [cellSpacing](#) property. Adding spacing between the panels will make the layout effective and provides a clear data representation.

The following sample demonstrates the usage of the [cellSpacing](#) property which helps in a neat and clear representation of a data.

INDEX.JS

```

// initialize dashboardlayout component
var dashboard = new ej.layouts.DashboardLayout({
    cellSpacing: [20, 20],
    columns: 5,
    panels: [{ "sizeX": 1, "sizeY": 1, "row": 0, "col": 0, content:'<div
class="content">0</div>' },
    { "sizeX": 3, "sizeY": 2, "row": 0, "col": 1, content:'<div
class="content">1</div>' },
    { "sizeX": 1, "sizeY": 3, "row": 0, "col": 4, content:'<div
class="content">2</div>' },
    { "sizeX": 1, "sizeY": 1, "row": 1, "col": 0, content:'<div
class="content">3</div>' },
    { "sizeX": 2, "sizeY": 1, "row": 2, "col": 0, content:'<div
class="content">4</div>' },
    { "sizeX": 1, "sizeY": 1, "row": 2, "col": 2, content:'<div
class="content">5</div>' },
    { "sizeX": 1, "sizeY": 1, "row": 2, "col": 3, content:'<div
class="content">6</div>' }
    ]
});
// render initialized dashboardlayout
dashboard.appendTo('#dashboard_layout');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>

```

```

<title>Essential JS 2 DashboardLayout </title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Essential JS 2 DashboardLayout
Component">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container">
    <!--element which is going to render the dashboardlayout-->
    <div id="dashboard_layout"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Graphical representation of layout

These cells combinedly will form a grid-structured layout which will be hidden initially. This grid structured layout can be made visible by enabling the [showGridLines](#) property which clearly pictures the cells split-up within the layout. These gridlines will be helpful in panels sizing and placement within the layout during initial designing of a dashboard.

In the following sample, the grid lines indicate the cells split-up of the layout and the data containers placed over these cells are known as panels.

INDEX.JS

```

// initialize dashboardlayout component
var dashboard = new ej.layouts.DashboardLayout({
  cellSpacing: [10, 10],
  showGridLines: true,
  columns: 5,
  panels: [

```



```

    { "sizeX": 3, "sizeY": 2, "row": 0, "col": 1, content: '<div
class="content">1</div>' },
    { "sizeX": 1, "sizeY": 3, "row": 0, "col": 4, content: '<div
class="content">2</div>' },
    { "sizeX": 1, "sizeY": 1, "row": 2, "col": 2, content: '<div
class="content">3</div>' },
    { "sizeX": 1, "sizeY": 1, "row": 2, "col": 3, content: '<div
class="content">4</div>' }
  ]
});
// render initialized dashboardlayout
dashboard.appendTo('#dashboard_layout');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DashboardLayout </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 DashboardLayout
Component">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container">
    <!--element which is going to render the dashboardlayout-->
    <div id="dashboard_layout"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Rendering component in right-to-left direction

It is possible to render the Dashboard Layout in right-to-left direction by setting the [enableRtl](#) API to true.

The following sample demonstrates Dashboard Layout in right-to-left direction.

INDEX.JS

```
// initialize dashboardlayout component
var dashboard = new ej.layouts.DashboardLayout({
  cellSpacing: [10, 10],
  enableRtl: true,
  columns: 5,
  panels: [{ 'sizeX': 1, 'sizeY': 1, 'row': 0, 'col': 0, header:
'<div>Panel 0</div>', content: '<div class="content"></div>',
  { 'sizeX': 3, 'sizeY': 2, 'row': 0, 'col': 1, header: '<div>Panel
1</div>', content: '<div class="content"></div>',
  { 'sizeX': 1, 'sizeY': 3, 'row': 0, 'col': 4, header: '<div>Panel
2</div>', content: '<div class="content"></div>',
  { 'sizeX': 1, 'sizeY': 1, 'row': 1, 'col': 0, header: '<div>Panel
3</div>', content: '<div class="content"></div>',
  { 'sizeX': 2, 'sizeY': 1, 'row': 2, 'col': 0, header: '<div>Panel
4</div>', content: '<div class="content"></div>',
  { 'sizeX': 1, 'sizeY': 1, 'row': 2, 'col': 2, header: '<div>Panel
5</div>', content: '<div class="content"></div>',
  { 'sizeX': 1, 'sizeY': 1, 'row': 2, 'col': 3, header: '<div>Panel
6</div>', content: '<div class="content"></div>'}]
});
// render initialized dashboardlayout
dashboard.appendTo('#dashboard_layout');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DashboardLayout </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 DashboardLayout
Component">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container">
    <div id="dashboard_layout"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
```

```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Panels

Position sizing of panels in `##Platform_Name##` Dashboard layout control

Panels are the basic building blocks of the dashboard layout component. They act as a container for the data to be visualized or presented. These panels can be positioned or resized for effective presentation of the data.

The below table represents all the available panel properties and the corresponding functionalities

PanelObject	Description
---	---
id	Specifies the id value of the panel.
row	Specifies the row value in which the panel to be placed.
col	Specifies the column value in which the panel to be placed.
sizeX	Specifies the width of the panel in cells count.
sizeY	Specifies the height of the panel in cells count.
minSizeX	Specifies the minimum width of the panel in cells count.
minSizeY	Specifies the minimum height of the panel in cells count.
maxSizeX	Specifies the maximum width of the panel in cells count.
maxSizeY	Specifies the maximum height of the panel in cells count.
header	Specifies the header template of the panel.
content	Specifies the content template of the panel.
cssClass	Specifies the CSS class name that can be appended with each panel element.

Positioning of panels

The panels within the layout can be easily positioned or ordered using the `row` and `col` properties of the panels. Positioning of panels will be beneficial to represent the data in any desired order.

The following sample demonstrates the positioning of panels within the dashboard layout using the `row`, `column` properties of the panels.

INDEX.JS

```
// initialize dashboardlayout component
let dashboard = new ej.layouts.DashboardLayout({
  cellSpacing: [20, 20],
  columns: 3,
  panels: [
    { "row": 0, "col": 0, content: '<div class="content">1</div>' },
    { "row": 0, "col": 1, content: '<div class="content">2</div>' },
    { "row": 0, "col": 2, content: '<div class="content">3</div>' },
    { "row": 1, "col": 0, content: '<div class="content">4</div>' },
  ]
});
```

```

    { "row": 1, "col": 1, content: '<div class="content">5</div>' },
    { "row": 1, "col": 2, content: '<div class="content">6</div>' }
  ]
});
// render initialized dashboardlayout
dashboard.appendTo('#dashboard_default');
```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>Essential JS 2 DashboardLayout </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 DashboardLayout
Component">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container">
    <!--element which is going to render the dashboardlayout-->
    <div id="dashboard_default"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>
```

Sizing of panels

A panel's size can be varied easily by defining the `sizeX` and `sizeY` properties. The `sizeX` property defines the width and `sizeY` property defines height of a panel in cells count. These properties will be helpful in designing a dashboard, where the content of each panel may vary in size.

The following sample demonstrates the sizing of panels within the dashboard layout using the `sizeX` and `sizeY` properties of the panels.

INDEX.JS

```

// initialize dashboardlayout component
let dashboard = new ej.layouts.DashboardLayout({
```

```

    cellSpacing: [20, 20],
    columns: 5,
    panels: [{ "sizeX": 1, "sizeY": 1, "row": 0, "col": 0, content: '<div
class="content">0</div>' },
    { "sizeX": 3, "sizeY": 2, "row": 0, "col": 1, content: '<div
class="content">1</div>' },
    { "sizeX": 1, "sizeY": 3, "row": 0, "col": 4, content: '<div
class="content">2</div>' },
    { "sizeX": 1, "sizeY": 1, "row": 1, "col": 0, content: '<div
class="content">3</div>' },
    { "sizeX": 2, "sizeY": 1, "row": 2, "col": 0, content: '<div
class="content">4</div>' },
    { "sizeX": 1, "sizeY": 1, "row": 2, "col": 2, content: '<div
class="content">5</div>' },
    { "sizeX": 1, "sizeY": 1, "row": 2, "col": 3, content: '<div
class="content">6</div>' }
    ]
  });
  // render initialized dashboardlayout
  dashboard.appendTo('#dashboard default');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DashboardLayout </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 DashboardLayout
Component">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container">
    <!--element which is going to render the dashboardlayout-->
    <div id="dashboard_default"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Setting header of panels in ##Platform_Name## Dashboard layout control

The Dashboard layout component is mostly used to represent the data used for monitoring or managing a process. These data or any HTML template can be placed as the content of a panel using the `content` property. Also, word or phrase that summarize about the panel's content can be added as the header on the top of each panel using the `header` property of the panel.

The following sample demonstrates how to add content for each panel using the header and content properties of the panels.

INDEX.JS

```
// initialize dashboardlayout component
var dashboard = new ej.layouts.DashboardLayout({
  cellSpacing: [10, 10],
  columns: 6,
  panels: [{ 'id': 'Panel0', 'sizeX': 1, 'sizeY': 1, 'row': 0, 'col': 0,
  header: '<div>Panel 0</div>', content: '<div class="content">Panel
Content<div>',
    { 'id': 'Panel1', 'sizeX': 3, 'sizeY': 2, 'row': 0, 'col': 1,
  header: '<div>Panel 1</div>', content: '<div class="content">Panel
Content<div>',
    { 'id': 'Panel2', 'sizeX': 1, 'sizeY': 3, 'row': 0, 'col': 4,
  header: '<div>Panel 2</div>', content: '<div class="content">Panel
Content<div>',
    { 'id': 'Panel3', 'sizeX': 1, 'sizeY': 1, 'row': 1, 'col': 0,
  header: '<div>Panel 3</div>', content: '<div class="content">Panel
Content<div>',
    { 'id': 'Panel4', 'sizeX': 2, 'sizeY': 1, 'row': 2, 'col': 0,
  header: '<div>Panel 4</div>', content: '<div class="content">Panel
Content<div>',
    { 'id': 'Panel5', 'sizeX': 1, 'sizeY': 1, 'row': 2, 'col': 2,
  header: '<div>Panel 5</div>', content: '<div class="content">Panel
Content<div>',
    { 'id': 'Panel6', 'sizeX': 1, 'sizeY': 1, 'row': 2, 'col': 3,
  header: '<div>Panel 6</div>', content: '<div class="content">Panel
Content<div>'
  ]
});
// render initialized dashboardlayout
dashboard.appendTo('#dashboard_default');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DashboardLayout </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 DashboardLayout
Component">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container">
    <div id="dashboard_default"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Placing components as content of panels

In a dashboard, components like the chart, grids, maps, gauge etc. can be used to present a complex data. Any such components can be placed as the panel content by assigning the corresponding component element as the **content** of the panel.

The following sample demonstrates how to add ej2-chart components as the **content** for each panel in the dashboard layout component.

INDEX.JS

```

// initialize dashboardlayout component
let dashboard = new ej.layouts.DashboardLayout({
  cellSpacing: [10, 10],
  columns: 6,
  panels: [{ 'id': 'Panel1', 'sizeX': 3, 'sizeY': 2, 'row': 0, 'col': 0,
    header: '<div class="header"> Product usage ratio </div>', content: '<div
    id="pie"><div>',
    { 'id': 'Panel2', 'sizeX': 3, 'sizeY': 2, 'row': 0, 'col': 3, header: '<div
    class="header"> Last year Sales Comparison </div>', content: '<div
    id="column"><div>' },
    { 'id': 'Panel3', 'sizeX': 3, 'sizeY': 2, 'row': 0, 'col': 3, header: '<div
    class="header"> Mobile browsers usage </div>', content: '<div
    id="pie1"><div>' },
    { 'id': 'Panel4', 'sizeX': 3, 'sizeY': 2, 'row': 1, 'col': 0, header: '<div
    class="header"> Sales increase percentage </div>', content: '<div
    id="line"><div>' }
  ]
});
// render initialized dashboardlayout
dashboard.appendTo('#dashboard_default');
let chartData = [
  { month: 'Jan', sales: 35 }, { month: 'Feb', sales: 28 },
  { month: 'Mar', sales: 34 }, { month: 'Apr', sales: 32 },
  { month: 'May', sales: 40 }, { month: 'Jun', sales: 32 },

```

```

        { month: 'Jul', sales: 35 }, { month: 'Aug', sales: 55 },
        { month: 'Sep', sales: 38 }, { month: 'Oct', sales: 30 },
        { month: 'Nov', sales: 25 }, { month: 'Dec', sales: 32 }
    ];
    let chart = new ej.charts.Chart({
        primaryXAxis: {
            valueType: 'Category'
        },
        series:[{
            dataSource: chartData,
            xName: 'month',
            yName: 'sales',
            type: 'Column'
        }],
        height:"162px"
    }, '#column');
    let lineData = [
        { x: 2013, y: 28 }, { x: 2014, y: 25 }, { x: 2015, y: 26 }, { x: 2016,
        y: 27 },
        { x: 2017, y: 32 }, { x: 2018, y: 35 },
    ];
    let linechart = new ej.charts.Chart({
        series:[{
            dataSource: lineData,
            xName: 'x', yName: 'y',
            //Series type as line
            type: 'Line'
        }],
        height:"162px"
    }, '#line');
    let accChart = new ej.charts.AccumulationChart({
        series: [
            {
                dataSource: [{ x: 'TypeScript', y: 13, text: 'TS 13%' }, { x:
                'React', y: 12.5, text: 'Reat 12.5%' }, { x: 'MVC', y: 12, text: 'MVC 12%'
                }, { x: 'Core', y: 12.5, text: 'Core 12.5%' }, { x: 'Vue', y: 10, text: 'Vue
                10%' }, { x: 'Angular', y: 40, text: 'Angular 40%' }],
                xName: 'x',
                yName: 'y',
                innerRadius: "20%"
            },
            {
                tooltip:{enable: true},
                height:"162px"
            }
        ],
        '#pie');
    let piechart = new ej.charts.AccumulationChart({
        // Initialize the chart series
        series: [
            {
                dataSource: [
                    { 'x': 'Chrome', y: 37, text: '37%' }, { 'x': 'UC
                    Browser', y: 17, text: '17%' },
                    { 'x': 'iPhone', y: 19, text: '19%' },
                    { 'x': 'Others', y: 4, text: '4%' }, { 'x': 'Opera', y:
                    11, text: '11%' },
                    { 'x': 'Android', y: 12, text: '12%' }
                ],
                dataLabel: {

```



```

        visible: true, position: 'Inside', name: 'text', font: {
fontWeight: '600' }
        },
        radius: '70%', xName: 'x', yName: 'y', name: 'Browser'
    }
    ],
    center: { x: '50%', y: '50%' },
    enableSmartLabels: true,
    height: "162px",
    enableAnimation: false,
    legendSettings: { visible: false },
    // Initialize tht tooltip
    tooltip: { enable: true, format: '${point.x} : <b>${point.y}%</b>'
},
}, '#piel');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>

    <title>Essential JS 2 DashboardLayout </title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Essential JS 2 DashboardLayout
Component">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="//cdn.syncfusion.com/ej2/21.2.3/ej2-base/styles/material.css"
rel="stylesheet">
    <link href="//cdn.syncfusion.com/ej2/21.2.3/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="//cdn.syncfusion.com/ej2/21.2.3/ej2-
layouts/styles/material.css" rel="stylesheet">
    <link href="//cdn.syncfusion.com/ej2/21.2.3/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="//cdn.syncfusion.com/ej2/21.2.3/ej2-
circulargauge/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <div id="container">
        <div id="dashboard_default"></div>
    </div>

    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>

```

```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Add remove panels in ##Platform_Name## Dashboard layout control

In real-time cases, the data being presented within the dashboard need to be updated frequently which includes adding or removing the data dynamically within the dashboard. This can be easily achieved by using the [addPanel](#) and [removePanel](#) public methods of the component.

Add or remove panels dynamically

Panels can be added dynamically by using the [addPanel](#) public method by passing the [panel](#) property as parameter. Also, they can be removed dynamically by using the [removePanel](#) public method by simply passing the [panel id](#) value as a parameter.

It is also possible to remove all the panels in a Dashboard Layout by calling [removeAll](#) method.

```
`js
```

```
dashboard.removeAll();
```

```
,
```

The following sample demonstrates how to add and remove the panels dynamically in the dashboard layout component. Here, panels can be added in any desired position of required size by selecting them in the numeric boxes and clicking add button and remove them by selecting the id of the panel.

INDEX.JS

```
// initialize dashboardlayout component
var dashboard = new ej.layouts.DashboardLayout({
  cellSpacing: [10, 10],
  columns: 5,
  panels: [{ 'id': 'Panel0', 'sizeX': 1, 'sizeY': 1, 'row': 0, 'col': 0,
  content: '<div class="content">0</div>' },
  { 'id': 'Panel1', 'sizeX': 3, 'sizeY': 2, 'row': 0, 'col': 1, content: '<div
class="content">1</div>' },
  { 'id': 'Panel2', 'sizeX': 1, 'sizeY': 3, 'row': 0, 'col': 4, content: '<div
class="content">2</div>' },
  { 'id': 'Panel3', 'sizeX': 1, 'sizeY': 1, 'row': 1, 'col': 0, content: '<div
class="content">3</div>' },
  { 'id': 'Panel4', 'sizeX': 2, 'sizeY': 1, 'row': 2, 'col': 0, content: '<div
class="content">4</div>' },
  { 'id': 'Panel5', 'sizeX': 1, 'sizeY': 1, 'row': 2, 'col': 2, content: '<div
class="content">5</div>' },
  { 'id': 'Panel6', 'sizeX': 1, 'sizeY': 1, 'row': 2, 'col': 3, content: '<div
class="content">6</div>' }
  ]
});
// render initialized dashboardlayout
dashboard.appendTo('#dashboard_layout');
var count = 7;
var data = ["Panel0", "Panel1", "Panel2", "Panel3", "Panel4", "Panel5",
"Panel6"];
var sizeX = new ej.inputs.NumericTextBox ({
  placeholder: 'Ex: 1',
  floatLabelType: 'Never',
  value: 1,
```

```

        min:1,
        max: 5
    });
    sizeX.appendTo('#sizeX');
    var idValue = new ej.dropdowns.DropDownList ({
        dataSource: data
    });
    idValue.appendTo("#value");
    var sizeY = new ej.inputs.NumericTextBox ({
        //set the data to dataSource property
        placeholder: 'Ex: 1',
        floatLabelType: 'Never',
        value: 1,
        min:1,
        max: 5
    });
    sizeY.appendTo('#sizeY');
    var row = new ej.inputs.NumericTextBox ({
        //set the data to dataSource property
        placeholder: 'Ex: 1',
        floatLabelType: 'Never',
        value: 0,
        min:0,
        max: 5
    });
    row.appendTo('#row');
    var column = new ej.inputs.NumericTextBox ({
        //set the data to dataSource property
        placeholder: 'Ex: 1',
        floatLabelType: 'Never',
        value: 0,
        min:0,
        max: 4
    });
    column.appendTo('#column');
    document.getElementById('add').onclick = function() {
        var panel = {
            id: "Panel"+ count.toString(),
            sizeX: sizeX.value,
            sizeY: sizeY.value,
            row: row.value,
            col: column.value,
            content: "<div class='content'>" + count + "</div>"
        }
        dashboard.addPanel(panel);
        count = count + 1;
        (idValue.dataSource).push(panel.id);
        idValue.refresh();
    };
    document.getElementById('remove').onclick = function() {
        dashboard.removePanel(idValue.value.toString());

        (idValue.dataSource).splice((idValue.dataSource).indexOf(idValue.value.toString()), 1);
        idValue.refresh();
        idValue.value = null;
    }
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DashboardLayout </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 DashboardLayout
Component">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container">
    <div class="inline" id="control">
      <div id="dashboard_layout"></div>
    </div>
    <div class="inline" id="properties">
      <table>
        <tbody><tr>
          <td>SizeX</td>
          <td> <input id="sizeX"></td>
        </tr>
        <tr>
          <td>SizeX</td>
          <td> <input id="sizeY"></td>
        </tr>
        <tr>
          <td>Row</td>
          <td> <input id="row"></td>
        </tr>
        <tr>
          <td>Column</td>
          <td> <input id="column"></td>
        </tr>
        <tr>
          <td> </td>
          <td>
            <button id="add" class="e-btn e-flat e-
outline">Add</button>

```

```

        </td>
      </tr>
    </tbody></table>
    <table>
      <tbody><tr>
        <td>Id</td>
        <td> <input id="value"></td>
      </tr>
      <tr>
        <td> </td>
        <td>
          <button id="remove" class="e-btn e-flat e-
danger">Remove</button>
        </td>
      </tr>
    </tbody></table>
  </div>
</div>

<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Interaction With Panels

Dragging moving of panels in ##Platform_Name## Dashboard layout control

The Dashboard Layout component is provided with dragging functionality to drag and reorder the panels within the layout. While dragging a panel, a holder will be highlighted below the panel indicating the panel placement on panel drop. This helps the user to decide whether to place the panel in the current position or revert to previous position without disturbing the layout.

If one or more panels collide while dragging, then the colliding panels will be pushed towards left or right or top or bottom direction where an adaptive space for the collided panel is available. The position changes of these collided panels will be updated dynamically during dragging of a panel so the user can conclude whether to place the panel in the current position or not.

While dragging a panel in Dashboard layout the following dragging events will be triggered,

- [dragStart](#) - Triggers when panel drag starts
- [drag](#) - Triggers when panel is being dragged
- [dragStop](#) - Triggers when panel drag stops

The following sample demonstrates dragging and pushing of panels. Here, for e.g. While dragging the panel 0 over panel 1, these panels get collided and push the panel 1 towards the feasible direction so that panel 0 gets placed in the panel 1 position.

INDEX.JS

```
// initialize dashboardlayout component
```

```

var dashboard = new ej.layouts.DashboardLayout({
  cellSpacing: [10, 10],
  columns: 5,
  panels: [{ 'sizeX': 1, 'sizeY': 1, 'row': 0, 'col': 0, content: '<div
class="content">0</div>' },
  { 'sizeX': 3, 'sizeY': 2, 'row': 0, 'col': 1, content: '<div
class="content">1</div>' },
  { 'sizeX': 1, 'sizeY': 3, 'row': 0, 'col': 4, content: '<div
class="content">2</div>' },
  { 'sizeX': 1, 'sizeY': 1, 'row': 1, 'col': 0, content: '<div
class="content">3</div>' },
  { 'sizeX': 2, 'sizeY': 1, 'row': 2, 'col': 0, content: '<div
class="content">4</div>' },
  { 'sizeX': 1, 'sizeY': 1, 'row': 2, 'col': 2, content: '<div
class="content">5</div>' },
  { 'sizeX': 1, 'sizeY': 1, 'row': 2, 'col': 3, content: '<div
class="content">6</div>' } ]
});
// render initialized dashboardlayout
dashboard.appendTo('#dashboard layout');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DashboardLayout </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 DashboardLayout
Component">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container">
    <!--element which is going to render the dashboardlayout-->
    <div id="dashboard_layout"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Customizing the dragging handler

Initially, the complete panel will act as the handler for dragging the panel such that the dragging action occurs on clicking anywhere over a panel. However, this dragging handler for the panels can be customized using the [draggableHandle](#) property to restrict the dragging action within a particular element in the panel.

The following sample demonstrates customizing the dragging handler of the panels where dragging action of panel occurs only with the header of the panel.

INDEX.JS

```
// initialize dashboardlayout component
var dashboard = new ej.layouts.DashboardLayout({
  cellSpacing: [10, 10],
  columns: 6,
  draggableHandle: '.e-panel-header',
  panels: [{ 'id': 'Panel1', 'sizeX': 3, 'sizeY': 2, 'row': 0, 'col': 0,
    header: '<div class="header"> Product usage ratio </div><span class="handler e-icons burg-icon"></span>', content: '<div id="pie"><div>' },
    { 'id': 'Panel2', 'sizeX': 3, 'sizeY': 2, 'row': 0, 'col': 3, header: '<div
class="header"> Last year Sales Comparison </div> <span class="handler e-
icons burg-icon"></span>', content: '<div id="column"><div>' },
    { 'id': 'Panel3', 'sizeX': 3, 'sizeY': 2, 'row': 0, 'col': 3, header: '<div
class="header"> Mobile browsers usage </div><span class="handler e-icons
burg-icon"></span>', content: '<div id="pie1"><div>' },
    { 'id': 'Panel4', 'sizeX': 3, 'sizeY': 2, 'row': 1, 'col': 0, header: '<div
class="header"> Sales increase percentage </div><span class="handler e-icons
burg-icon"></span>', content: '<div id="line"><div>' }
  ]
});
// render initialized dashboardlayout
dashboard.appendTo('#dashboard_default');
var chartData = [
  { month: 'Jan', sales: 35 }, { month: 'Feb', sales: 28 },
  { month: 'Mar', sales: 34 }, { month: 'Apr', sales: 32 },
  { month: 'May', sales: 40 }, { month: 'Jun', sales: 32 },
  { month: 'Jul', sales: 35 }, { month: 'Aug', sales: 55 },
  { month: 'Sep', sales: 38 }, { month: 'Oct', sales: 30 },
  { month: 'Nov', sales: 25 }, { month: 'Dec', sales: 32 }
];
var chart = new ej.charts.Chart({
  primaryXAxis: {
    valueType: 'Category'
  },
  series: [{
    dataSource: chartData,
    xName: 'month',
    yName: 'sales',
    type: 'Column'
  }],
  height: "162px"
}, '#column');
var lineData = [
  { x: 2013, y: 28 }, { x: 2014, y: 25 }, { x: 2015, y: 26 }, { x: 2016,
y: 27 },
  { x: 2017, y: 32 }, { x: 2018, y: 35 },
];
```

```

var linechart = new ej.charts.Chart({
  series:[{
    dataSource: lineData,
    xName: 'x', yName: 'y',
    //Series type as line
    type: 'Line'
  }],
  height:"162px"
}, '#line');
var accChart = new ej.charts.AccumulationChart({
  series: [
    {
      dataSource: [{ x: 'TypeScript', y: 13, text: 'TS 13%' }, { x:
'React', y: 12.5, text: 'Reat 12.5%' }, { x: 'MVC', y: 12, text: 'MVC 12%'
},{ x: 'Core', y: 12.5, text: 'Core 12.5%' }, { x: 'Vue', y: 10, text: 'Vue
10%' }, { x: 'Angular', y: 40, text: 'Angular 40%' } ],
      xName: 'x',
      yName: 'y',
      innerRadius: "20%"
    }],
    tooltip:{enable: true},
    height:"162px"
}, '#pie');
var piechart = new ej.charts.AccumulationChart({
  // Initialize the chart series
  series: [
    {
      dataSource: [
        { 'x': 'Chrome', y: 37, text: '37%' }, { 'x': 'UC
Browser', y: 17, text: '17%' },
        { 'x': 'iPhone', y: 19, text: '19%' },
        { 'x': 'Others', y: 4, text: '4%' }, { 'x': 'Opera', y:
11, text: '11%' },
        { 'x': 'Android', y: 12, text: '12%' }
      ],
      dataLabel: {
        visible: true, position: 'Inside', name: 'text', font: {
fontWeight: '600' }
      },
      radius: '70%', xName: 'x', yName: 'y', name: 'Browser'
    }
  ],
  center: { x: '50%', y: '50%' },
  enableSmartLabels: true,
  height:"162px",
  enableAnimation: false,
  legendSettings: { visible: false },
  // Initialize tht tooltip
  tooltip: { enable: true, format: '$ {point.x} : <b>${point.y}%</b>'
},
}, '#pie1');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DashboardLayout </title>

```



```

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Essential JS 2 DashboardLayout
Component">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
circulargauge/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container">
    <div id="dashboard_default"></div>
  </div>

<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Disable dragging of panels

By default, the dragging of panels is enabled in Dashboard Layout. It can also be disabled with the help of [allowDragging](#) API. Setting [allowDragging](#) to false disables the dragging functionality in Dashboard Layout.

The following sample demonstrates Dashboard Layout with dragging support disabled.

INDEX.JS

```

// initialize dashboardlayout component
var dashboard = new ej.layouts.DashboardLayout({
  cellSpacing: [10, 10],
  allowDragging: false,
  columns: 5,
  panels: [{ 'sizeX': 1, 'sizeY': 1, 'row': 0, 'col': 0, content: '<div
class="content">0</div>' },
  { 'sizeX': 3, 'sizeY': 2, 'row': 0, 'col': 1, content: '<div
class="content">1</div>' },

```

```

    {'sizeX': 1, 'sizeY': 3, 'row': 0, 'col': 4, content:'<div
class="content">2</div>'},
    {'sizeX': 1, 'sizeY': 1, 'row': 1, 'col': 0, content:'<div
class="content">3</div>'},
    {'sizeX': 2, 'sizeY': 1, 'row': 2, 'col': 0, content:'<div
class="content">4</div>'},
    {'sizeX': 1, 'sizeY': 1, 'row': 2, 'col': 2, content:'<div
class="content">5</div>'},
    {'sizeX': 1, 'sizeY': 1, 'row': 2, 'col': 3, content:'<div
class="content">6</div>'}]]
  });
  // render initialized dashboardlayout
  dashboard.appendTo('#dashboard_layout');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DashboardLayout </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 DashboardLayout
Component">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container">
    <div id="dashboard_layout"></div>
  </div>

  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Moving panels in ##Platform_Name## Dashboard layout control

Other than drag and drop, it is possible to move the panels in Dashboard Layout programmatically. This can be achieved using [movePanel](#) method. The method is invoked as follows,

```
`js
movePanel(id, row, col)
`
```

Where,

- id - ID of the panel which needs to be moved.
- row - New row position for moving the panel.
- col - New column position for moving the panel.

Each time a panel's position is changed(Programatically or through UI interaction), the Dashboard Layout's [change](#) event will be triggered.

The following sample demonstrates moving a panel programmatically to a new position in the Dashboard Layout's [created](#) event.

INDEX.JS

```
// initialize dashboardlayout component
var dashboard = new ej.layouts.DashboardLayout({
  cellSpacing: [10, 10],
  //Dashboard Layout's created event
  created: onCreate,
  //Dashboard Layout's change event
  change: onChange,
  columns: 5,
  panels: [{ 'sizeX': 1, 'sizeY': 1, 'row': 0, 'col': 0, content: '<div
class="content">0</div>' },
    { 'sizeX': 3, 'sizeY': 2, 'row': 0, 'col': 1, content: '<div
class="content">1</div>' },
    { 'sizeX': 1, 'sizeY': 3, 'row': 0, 'col': 4, content: '<div
class="content">2</div>' },
    { 'sizeX': 1, 'sizeY': 1, 'row': 1, 'col': 0, content: '<div
class="content">3</div>' },
    { 'sizeX': 2, 'sizeY': 1, 'row': 2, 'col': 0, content: '<div
class="content">4</div>' },
    { 'sizeX': 1, 'sizeY': 1, 'row': 2, 'col': 2, content: '<div
class="content">5</div>' },
    { 'sizeX': 1, 'sizeY': 1, 'row': 2, 'col': 3, content: '<div
class="content">6</div>' } ]
});
// render initialized dashboardlayout
dashboard.appendTo('#dashboard_layout');
//Dashboard Layout's created event function
function onCreate(args) {
  // movePanel("id", row, col)
  this.movePanel("layout_0", 1, 0);
}
//Dashboard Layout's change event function
function onChange(args) {
  console.log("Change event triggered");
}
```

```
}

```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DashboardLayout </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 DashboardLayout
Component">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container">
    <!--element which is going to render the dashboardlayout-->
    <div id="dashboard_layout"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Resizing of panels in ##Platform_Name## Dashboard layout control

The DashboardLayout component is also provided with the panel resizing functionality which can be enabled or disabled using the [allowResizing](#) property. This functionality allows to resize the panels dynamically through UI interactions using the resizing handlers which controls the panel resizing in various directions.

Initially, the panels can be resized only in south-east direction. However, panels can also be resized in east, west, north, south and south-west directions by defining the required directions with [resizableHandles](#) property.

On resizing a panel in Dashboard layout the following events will be triggered,

- [resizeStart](#) - Triggers when panel resize starts
- [resize](#) - Triggers when panel is being resized
- [resizeStop](#) - Triggers when panel resize stops

The following sample demonstrates how to enable and disable the resizing of panels in the DashboardLayout component in different directions.

INDEX.JS

```
// initialize dashboardlayout component
var dashboard = new ej.layouts.DashboardLayout({
  cellSpacing: [10, 10],
  allowResizing: true,
  columns: 5,
  //Dashboard Layout's resizestart event
  resizeStart: onResizeStart,
  //Dashboard Layout's resize event
  resize: onResize,
  //Dashboard Layout's resizestop event
  resizeStop: onResizeStop,
  resizableHandles: ['e-south-east', 'e-east', 'e-west', 'e-north', 'e-south'],
  panels: [{ 'sizeX': 1, 'sizeY': 1, 'row': 0, 'col': 0, content: '<div class="content">0</div>' },
    { 'sizeX': 3, 'sizeY': 2, 'row': 0, 'col': 1, content: '<div class="content">1</div>' },
    { 'sizeX': 1, 'sizeY': 3, 'row': 0, 'col': 4, content: '<div class="content">2</div>' },
    { 'sizeX': 1, 'sizeY': 1, 'row': 1, 'col': 0, content: '<div class="content">3</div>' },
    { 'sizeX': 2, 'sizeY': 1, 'row': 2, 'col': 0, content: '<div class="content">4</div>' },
    { 'sizeX': 1, 'sizeY': 1, 'row': 2, 'col': 2, content: '<div class="content">5</div>' },
    { 'sizeX': 1, 'sizeY': 1, 'row': 2, 'col': 3, content: '<div class="content">6</div>' } ]
});
// render initialized dashboardlayout
dashboard.appendTo('#dashboard_default');
//Dashboard Layout's resizestart event function
function onResizeStart(args) {
  console.log("Resize start");
}
//Dashboard Layout's resize event function
function onResize(args) {
  console.log("Resizing");
}
//Dashboard Layout's resizestop event function
function onResizeStop(args) {
  console.log("Resize stop");
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DashboardLayout </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 DashboardLayout Component">
```

```

<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container">
    <!--element which is going to render the dashboardlayout-->
    <div id="dashboard_default"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Resizing panels programmatically

The Dashboard Layout panels can also be resized programmatically by using [resizePanel](#) method. The method is invoked as follows,

```
`js
```

```
resizePanel(id, sizeX, sizeY)
```

```
,
```

Where,

- id - ID of the panel which needs to be resized.
- sizeX - New panel width in cells count for resizing the panel.
- sizeY - New panel height in cells count for resizing the panel.

The following sample demonstrates resizing panels programmatically in the Dashboard Layout's [created](#) event.

INDEX.JS

```

// initialize dashboardlayout component
var dashboard = new ej.layouts.DashboardLayout({
  cellSpacing: [10, 10],
  //Dashboard Layout's created event
  created: onCreate,
  columns: 5,
  panels: [{ 'sizeX': 1, 'sizeY': 1, 'row': 0, 'col': 0, content: '<div
class="content">0</div>' },

```

```

    {'sizeX': 3, 'sizeY': 2, 'row': 0, 'col': 1, content: '<div
class="content">1</div>'},
    {'sizeX': 1, 'sizeY': 3, 'row': 0, 'col': 4, content: '<div
class="content">2</div>'},
    {'sizeX': 1, 'sizeY': 1, 'row': 1, 'col': 0, content: '<div
class="content">3</div>'},
    {'sizeX': 2, 'sizeY': 1, 'row': 2, 'col': 0, content: '<div
class="content">4</div>'},
    {'sizeX': 1, 'sizeY': 1, 'row': 2, 'col': 2, content: '<div
class="content">5</div>'},
    {'sizeX': 1, 'sizeY': 1, 'row': 2, 'col': 3, content: '<div
class="content">6</div>'}]]
    });
    // render initialized dashboardlayout
    dashboard.appendTo('#dashboard_layout');
    //Dashboard Layout's created event function
    function onCreated(args) {
        // resizePanel("id", sizeX, sizeY)
        this.resizePanel("layout_4", 1, 1);
        this.resizePanel("layout_5", 2, 1);
    }

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 DashboardLayout </title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Essential JS 2 DashboardLayout
Component">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <div id="container">
        <!--element which is going to render the dashboardlayout-->
        <div id="dashboard_layout"></div>
    </div>
    <script>
    var ele = document.getElementById('container');
    if(ele) {
        ele.style.visibility = "visible";
    }
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

Floating of panels in ##Platform_Name## Dashboard layout control

The floating functionality of the component allows to effectively use the entire layout for the panel's placement. If the floating functionality is enabled, the panels within the layout get floated upwards automatically to occupy the empty cells available in previous rows. This functionality can be enabled or disabled using the [allowFloating](#) property of the component.

The following sample demonstrates how to enable or disable the floating of panels in the DashboardLayout component.

INDEX.JS

```
// initialize dashboardlayout component
var dashboard = new ej.layouts.DashboardLayout({
  cellSpacing: [10, 10],
  allowFloating: false,
  cellAspectRatio: 100/75,
  columns: 6,
  panels: [{ 'sizeX': 2, 'sizeY': 2, 'row': 1, 'col': 0, content: '<div
class="content">0</div>' },
  { 'sizeX': 2, 'sizeY': 2, 'row': 2, 'col': 2, content: '<div
class="content">1</div>' },
  { 'sizeX': 2, 'sizeY': 2, 'row': 3, 'col': 4, content: '<div
class="content">2</div>' } ]
});
// render initialized dashboardlayout
dashboard.appendTo('#dashboard_default');
var resetPanels = dashboard.serialize();
resetPanels[0].content = '<div class="content">0</div>';
resetPanels[1].content = '<div class="content">1</div>';
resetPanels[2].content = '<div class="content">2</div>';
var toggleBtn = new ej.buttons.Button({
  cssClass: "e-flat e-primary e-outline",
  content: "Enable Floating",
  isToggle: true
});
toggleBtn.appendTo("#toggle");
document.getElementById('toggle').onclick = function() {
  var panels = [];
  if (toggleBtn.content == "Disable Floating and Reset") {
    toggleBtn.content = 'Enable Floating';
    dashboard.allowFloating = false;
    dashboard.panels = resetPanels;
  } else {
    toggleBtn.content = 'Disable Floating and Reset';
    dashboard.allowFloating = true;
  }
};
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DashboardLayout </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
```



```

<meta name="description" content="Essential JS 2 DashboardLayout
Component">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container">
    <div class="inline" id="control">
      <div id="dashboard_default"></div>
    </div>
    <!--element which is going to render the dashboardlayout-->
    <div class="inline" id="properties">
      <button id="toggle"></button>
    </div>

  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Responsive adaptive in ##Platform_Name## Dashboard layout control

The control is provided with built-in responsive support, where panels within the layout get adjusted based on their parent element's dimensions to accommodate any resolution which relieves the burden of building responsive dashboards.

The dashboard layout is designed to automatically adapt with lower resolutions by transforming the entire layout into a stacked one so that the panels will be displayed in a vertical column. By default, whenever the screen resolution meets 600px or lower resolutions this layout transformation occurs. This transformation can be modified for any user defined resolution by defining the for the [mediaQuery](#) property of the component.

The following sample demonstrates the usage of [mediaQuery](#) property to turn out the layout into a stacked one in user defined resolution. Here, whenever, the window size reaches 700px or lesser, the layout becomes a stacked layout.

INDEX.JS

```
// initialize dashboardlayout component
```

```

var dashboard = new ej.layouts.DashboardLayout({
  cellSpacing: [20, 20],
  mediaQuery: 'max-width: 700px',
  columns: 5,
  panels: [{ "sizeX": 1, "sizeY": 1, "row": 0, "col": 0, content: '<div
class="content">0</div>' },
  { "sizeX": 3, "sizeY": 2, "row": 0, "col": 1, content: '<div
class="content">1</div>' },
  { "sizeX": 1, "sizeY": 3, "row": 0, "col": 4, content: '<div
class="content">2</div>' },
  { "sizeX": 1, "sizeY": 1, "row": 1, "col": 0, content: '<div
class="content">3</div>' },
  { "sizeX": 2, "sizeY": 1, "row": 2, "col": 0, content: '<div
class="content">4</div>' },
  { "sizeX": 1, "sizeY": 1, "row": 2, "col": 2, content: '<div
class="content">5</div>' },
  { "sizeX": 1, "sizeY": 1, "row": 2, "col": 3, content: '<div
class="content">6</div>' }
  ]
});
// render initialized dashboardlayout
dashboard.appendTo('#dashboard_default');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DashboardLayout </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 DashboardLayout
Component">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container">
    <!--element which is going to render the dashboardlayout-->
    <div id="dashboard_default"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
<script src="index.js" type="text/javascript"></script>

```

```
</body></html>
```

Save restore in ##Platform_Name## Dashboard layout control

The current layout structure of the Dashboard Layout component can be obtained and saved to construct another dashboard with same panel structure using the `serialize` public method of the component. This method returns the component's current panel setting which can be used to construct a dashboard with the same layout settings.

The following sample demonstrates how to save and restore the state of the panels using the `serialize` method. Here, the panel's settings are stored on the save button click and restored to the previously saved panel setting on clicking the restore button.

INDEX.JS

```
// initialize dashboardlayout component
var dashboard = new ej.layouts.DashboardLayout({
  cellSpacing: [20, 20],
  columns: 5,
  panels: [{ "sizeX": 1, "sizeY": 1, "row": 0, "col": 0, content: '<div
class="content">0</div>' },
  { "sizeX": 3, "sizeY": 2, "row": 0, "col": 1, content: '<div
class="content">1</div>' },
  { "sizeX": 1, "sizeY": 3, "row": 0, "col": 4, content: '<div
class="content">2</div>' },
  { "sizeX": 1, "sizeY": 1, "row": 1, "col": 0, content: '<div
class="content">3</div>' },
  { "sizeX": 2, "sizeY": 1, "row": 2, "col": 0, content: '<div
class="content">4</div>' },
  { "sizeX": 1, "sizeY": 1, "row": 2, "col": 2, content: '<div
class="content">5</div>' },
  { "sizeX": 1, "sizeY": 1, "row": 2, "col": 3, content: '<div
class="content">6</div>' }
],
  created: restorePanelModel
});
// render initialized dashboardlayout
dashboard.appendTo('#dashboard_default');
var restoreModel;
var saveBtn = new ej.buttons.Button({
  cssClass: "e-primary",
  content: "Save",
});
saveBtn.appendTo("#save");
var restoreBtn = new ej.buttons.Button({
  cssClass: "e-flat e-outline",
  content: "Restore",
});
restoreBtn.appendTo("#restore");
document.getElementById('save').onclick = function() {
  restorePanelModel();
};
document.getElementById('restore').onclick = function() {
  dashboard.panels = restoreModel;
};
function restorePanelModel() {
```

```

restoreModel = dashboard.serialize();
restoreModel[0].content = '<div class="content">0</div>';
restoreModel[1].content = '<div class="content">1</div>';
restoreModel[2].content = '<div class="content">2</div>';
restoreModel[3].content = '<div class="content">3</div>';
restoreModel[4].content = '<div class="content">4</div>';
restoreModel[5].content = '<div class="content">5</div>';
restoreModel[6].content = '<div class="content">6</div>';
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DashboardLayout </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 DashboardLayout
Component">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container">
    <div class="inline" id="control">
      <div id="dashboard_default"></div>
    </div>
    <!--element which is going to render the dashboardlayout-->
    <div class="inline" id="properties">
      <button id="save"></button>
      <button id="restore"></button>
    </div>

  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Style in ##Platform_Name## Dashboard layout control

The following content provides the exact CSS structure that can be used to modify the control's appearance based on the user preference.

Customizing the dashboard layout panel header

Use the following CSS to customize the dashboard layout panel header.

`

```
.e-dashboardlayout.e-control .e-panel .e-panel-container .e-panel-header {  
color: #754131;  
background-color: #c9e2f7;  
text-align: center;  
}
```

`

Customizing the dashboard layout panel content

Use the following CSS to customize the dashboard layout panel content.

`

```
.e-dashboardlayout.e-control .e-panel .e-panel-container .e-panel-content {  
background-color: #c9e2f7;  
padding: 50px;  
}
```

`

Customizing the dashboard layout panel resize icon

Use the following CSS to customize the dashboard layout resize icon.

`

```
.e-dashboardlayout.e-control .e-panel .e-panel-container .e-resize.e-double{  
color: #0378d5;  
font-size: 30px;  
height: 20px;  
width: 20px;  
}
```

`

Customizing the dashboard layout panel background

Use the following CSS to customize the dashboard layout panel background.

`

```
.e-dashboardlayout.e-control.e-responsive {  
background: #b3d3ed;
```

```
}
,
```

How To

Resize the panel dynamically in `##Platform_Name##` Dashboard layout control

In Dashboard Layout, the height of a panel is based on its width. While resizing the panel, the height and width should be changed.

To resize the height of a panel alone, the [resizePanel](#) method is used. In this case, the [cellAspectRatio](#) property configures the height of the cells based on the cell width to height ratio (cell width/cell height ratio) when the height will not be completely adjusted to `sizeY` value.

Refer to the following code snippet to determine the height of a panel.

```
`ts
```

```
let panelContent: HTMLElement = document.getElementById("panelContent");
```

```
let panelHeight: number = panelContent.offsetHeight;
```

```
,
```

INDEX.JS

```
// initialize dashboardlayout component
var dashboardObject = new ej.layouts.DashboardLayout({
    cellAspectRatio: 100/70,
    columns: 4,
    panels: [{
        id: 'panel0', 'sizeX': 1, 'sizeY': 1, 'row': 0, 'col': 0, header:
        '<div>Panel 0</div>',
        content: '<div class="content" id="panelContent">Place your
content here</div>'
    },
    {
        id: 'panel1', 'sizeX': 1, 'sizeY': 1, 'row': 0, 'col': 1,
        header: '<div>Panel 1</div>',
        content: '<div class="content" id="panelContent">Place your
content here</div>'
    },
    {
        id: 'panel2', 'sizeX': 1, 'sizeY': 1, 'row': 0, 'col': 2,
        header: '<div>Panel 2</div>',
        content: '<div class="content" id="panelContent">Place your
content here</div>'
    }
    ]
});
// render initialized dashboardlayout
dashboardObject.appendTo('#defaultLayout');
var btnInstance = new ej.buttons.Button({cssClass: "e-outline e-success"});
btnInstance.appendTo('#editbtn');

document.getElementById('editbtn').onclick = function() {
    var panelContent = document.getElementById("panelContent");
    var panelHeight = panelContent.offsetHeight;
    var panelWidth = panelContent.offsetWidth;
    var diff = Math.round(panelHeight/panelWidth);
```

```

dashboardObject.resizePanel('panel0', 1, diff);
dashboardObject.resizePanel('panel1', 1, diff);
dashboardObject.resizePanel('panel2', 1, diff);
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DashboardLayout </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 DashboardLayout
Component">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container">
    <div id="defaultLayout"></div>
    <button id="editbtn">Resize panel</button>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Accessibility in ##Platform_Name## Dashboard Layout component

The Dashboard Layout component followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Dashboard Layout component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support | |

| [Section 508](#) Support | ` |`

| Screen Reader Support | ` |`

| Right-To-Left Support | ` |`

| Color Contrast | ` |`

| Mobile Device Support | ` |`

| Keyboard Navigation Support | ` |`

| [Accessibility Checker](#) Validation | ` |`

| [Axe-core](#) Accessibility Validation | ` |`

`<style>`

```
.post .post-content img {  
display: inline-block;  
margin: 0.5em 0;  
}
```

`</style>`

`<div> - All features of the component meet the requirement.</div>`

`<div> - Some features of the component do not meet the requirement.</div>`

`<div> - The component does not meet the requirement.</div>`

WAI-ARIA attributes

The Dashboard Layout component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Dashboard Layout component:

| **Attributes** | **Purpose** |

| --- | --- |

| `role=list` | Indicates the role as a list for the Dashboard Layout element. |

| `role=listitem` | Indicates the role as a listitem for the Dashboard panels. |

| `role=presentation` | Indicates the role as a presentation for the table when the `showGridLines` property is enabled. |

| **aria-grabbed** | When the panel is chosen for dragging, the aria-grabbed attribute is set to "true." If it's set to "false," the element can be grabbed for drag-and-drop, but it won't be actively held. |

Keyboard interaction

Keyboard support is not applicable for the Dashboard Layout.

Ensuring accessibility

The Dashboard Layout component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Dashboard Layout component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Dashboard Layout component with accessibility tools.

See also

- [Accessibility in Syncfusion ##Platform_Name## components](#)

DataManager

Data binding in ##Platform_Name## Data control

DataManager supports both RESTful JSON data services binding and local JavaScript object array binding.

Local data binding

DataManager can be bound to local data source by assigning the array of JavaScript objects to the **json** property or simply passing them

to the constructor while instantiating. Now the JavaScript object array can be queried and manipulated.

INDEX.JS

```
var template =
'<tr><td>${OrderID}</td><td>${CustomerID}</td><td>${EmployeeID}</td></tr>';
var compiledFunction = ej.base.compile(template);
var result = new ej.data.DataManager(data).executeLocal(new
ej.data.Query().take(8));
var table = (document.getElementById('datatable'));
result.forEach((data) => {
    table.appendChild(compiledFunction(data)[0]);
});
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
        <table id="datatable" class="e-table">
            <thead>
                <tr><th>Order ID</th><th>Customer ID</th><th>Employee
ID</th></tr>
            </thead>
            <tbody>
            </tbody>
        </table>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Remote data binding

DataManager can be bound to remote data source by assigning service end point URL to the **url** property. With the provided **url**, the **DataManager** handles all communication with the data server with help of queries.

When querying data, the **DataManager** will convert the query object(**Query**) into server request after calling [executeQuery](#) and waits for the server response(JSON format).

INDEX.JS

```

var template =
'<tr><td>${OrderID}</td><td>${CustomerID}</td><td>${EmployeeID}</td></tr>';
var compiledFunction = ej.base.compile(template);
const SERVICE_URI =
'https://services.syncfusion.com/js/production/api/orders';
var table = (document.getElementById('datatable'));
new ej.data.DataManager({ url: SERVICE_URI }).executeQuery(new
ej.data.Query().take(8)).then((e) => {
    (e.result).forEach((data) => {
        table.appendChild(compiledFunction(data)[0]);
    });
});

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Grid"></div>
    <table id="datatable" class="e-table">
      <thead>

```

```

        <tr><th>Order ID</th><th>Customer ID</th><th>Employee
ID</th></tr>
        </thead>
        <tbody>
        </tbody>
    </table>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The queried data will not be cached locally unless offline mode is enabled.

See Also

- [Binding with OData service](#)
- [Binding with ODataV4 service](#)
- [Binding with Web API](#)
- [How to write custom adaptor](#)
- [How to work in offline mode](#)
- [How to send additional parameters](#)
- [How to add custom request headers](#)

Adaptors in ##Platform_Name## Data control

Each data source or remote service uses different way in accepting request and sending back the response. **DataManager** cannot anticipate every way a data source works. To tackle this problem the **DataManager** uses the adaptor concept to communicate with particular data source.

For local data sources, the role of the data adaptor is to query the JavaScript object array based on the **Query** object and manipulate them.

When comes with remote datasource, the data adaptor is used to send the request that the server can understand and process the server response.

The adaptor can be assigned using the **adaptor** property of the **DataManager**.

Json adaptor

JsonAdaptor is used to query and manipulate JavaScript object array.

INDEX.JS

```

var template =
'<tr><td>${OrderID}</td><td>${CustomerID}</td><td>${EmployeeID}</td></tr>';
var compiledFunction = ej.base.compile(template);
var result = new ej.data.DataManager(data).executeLocal(new
ej.data.Query().take(8));
var table = (document.getElementById('datatable'));
result.forEach((data) => {
    table.appendChild(compiledFunction(data)[0]);
});

```

```
});
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Grid"></div>
    <table id="datatable" class="e-table">
      <thead>
        <tr><th>Order ID</th><th>Customer ID</th><th>Employee
ID</th></tr>
      </thead>
      <tbody>
      </tbody>
    </table>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
```

```

}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Url adaptor

UrlAdaptor act as the base adaptor for interacting with remote data services. Most of the built-in adaptors are derived from the **UrlAdaptor**.

`ts

```

import { DataManager, Query, UrlAdaptor } from '@syncfusion/ej2-data';
const SERVICE_URI: string = 'https://services.syncfusion.com/js/production/api/UrlDataSource';
new DataManager({
url: SERVICE_URI,
adaptor: new UrlAdaptor
}).executeQuery(new Query().take(8)).then((e) => {
//e.result will contain the records
});
`

```

UrlAdaptor expects response as a JSON object with properties **result** and **count** which contains the collection of entities and the total number of records respectively.

The sample response object should be as follows,

```

{
"result": [{..}, {..}, {..}, ...],
"count": 67
}
`

```

OData adaptor

OData is standardized protocol for creating and consuming data. You can retrieve data from OData service using **DataManager**. The **ODataAdaptor** helps you to interact with OData service. You can refer to the following code example of remote Data binding using OData service.

INDEX.JS

```

var template =
'<tr><td>${OrderID}</td><td>${CustomerID}</td><td>${EmployeeID}</td></tr>'
var compiledFunction = ej.base.compile(template);
const SERVICE_URI =
'https://services.syncfusion.com/js/production/api/Orders';
var table = (document.getElementById('datatable'));

```

```

new ej.data.DataManager({ url: SERVICE_URI, adaptor: new
ej.data.ODataAdaptor })
    .executeQuery(new ej.data.Query().take(8))
    .then((e) => {
        (e.result.items).forEach((data) => {
            table.appendChild(compiledFunction(data)[0]);
        });
    });

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <div id="container">
        <div id="Grid"></div>
        <table id="datatable" class="e-table">
            <thead>
                <tr>
                    <th>Order ID</th>
                    <th>Customer ID</th>
                    <th>Employee ID</th>
                </tr>
            </thead>

```

```

        <tbody>
        </tbody>
    </table>
</div>
<script>
    var ele = document.getElementById('container');
    if (ele) {
        ele.style.visibility = "visible";
    }
</script>
<script src="index.js" type="text/javascript"></script>
</body>
</html>

```

By default, **ODataAdaptor** is used by **DataManager**.

ODataV4 adaptor

The ODataV4 is an improved version of OData protocols and the **DataManager** can also retrieve and consume OData v4 services. For more details on OData v4 Services, refer the [odata documentation](#). You can use the **ODataV4Adaptor** to interact with ODataV4 service.

INDEX.JS

```

var template =
'<tr><td>${OrderID}</td><td>${CustomerID}</td><td>${EmployeeID}</td></tr>'
var compiledFunction = ej.base.compile(template);
const SERVICE_URI =
'https://services.odata.org/V4/Northwind/Northwind.svc/Orders/';
var table = (document.getElementById('datatable'));
new ej.data.DataManager({ url: SERVICE_URI, adaptor: new
ej.data.ODataV4Adaptor })
    .executeQuery(new ej.data.Query().take(8))
    .then((e) => {
        (e.result).forEach((data) => {
            table.appendChild(compiledFunction(data)[0]);
        });
    });

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
        <table id="datatable" class="e-table">
            <thead>
                <tr><th>Order ID</th><th>Customer ID</th><th>Employee
ID</th></tr>
            </thead>
            <tbody>
            </tbody>
        </table>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Web API adaptor

You can use the **WebApiAdaptor** to interact with Web API created with OData endpoint. The **WebApiAdaptor** is extended from the **ODataAdaptor**. Hence to use **WebApiAdaptor**, the endpoint should understand the OData formatted queries send along with request.

To enable OData query option for Web API, please refer to the [documentation](#)

`ts

```

import { DataManager, Query, WebApiAdaptor } from '@syncfusion/ej2-data';
const SERVICE_URI: string = 'https://services.syncfusion.com/js/production/api/Orders';
new DataManager({
    url: SERVICE_URI,

```

```

adaptor: new WebApiAdaptor
}).executeQuery(new Query().take(8)).then((e) => {
//e.result will contain the records
});
`

```

WebApiAdaptor expects JSON response from the server and the response object should contain properties **Items** and **Count** whose values are collection of entities and total count of the entities respectively.

The sample response object should look like below.

```

{
Items: [{..}, {..}, {..}, ...],
Count: 830
}
`

```

WebMethod Adaptor

The **WebMethodAdaptor** is used to bind data source from remote services and code behind methods. It can be enabled in Grid using Adaptor property of DataManager as **WebMethodAdaptor**.

For every operations, an Fetch post will be send to the specified data service.

```

`ts
import { DataManager, Query, WebMethodAdaptor } from '@syncfusion/ej2-data';
let SERVICE_URI = 'Default.aspx/DataSource';
new DataManager({
url: SERVICE_URI,
adaptor: new WebMethodAdaptor
}).executeQuery(new Query().take(8)).then((e) => {
//e.result will contain the records
});
`

```

WebMethodAdaptor expects JSON response from the server and the response object should contain properties **result** and **count** whose values are collection of entities and total count of the entities respectively.

The sample response object should look like below.

```

`
{

```

```

result: [{..}, {..}, {..}, ...],
count: 830
}
,

```

The controller method's data parameter name must be `value`.

Custom Data Adaptor

The `CustomDataAdaptor` provides an option to send your own request to handle the data operations.

You can get the current action details inside the `getData` method of `CustomDataAdaptor` to build the request. Once the data is fetched from the service successfully, then the `onSuccess` method can be invoked to handle the further data processing. In failure case, invoke the `onFailure` method.

```
`ts
```

```

import { DataManager, Query, CustomDataAdaptor, FetchOption } from '@syncfusion/ej2-data';
const SERVICE_URI: string = 'http://controller.com/actions';
new DataManager({
  adaptor: new CustomDataAdaptor({
    getData: function (option: FetchOption) {
      let request: object;
      fetch(SERVICE_URI, {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json; charset=utf-8',
        },
      }).then((response) => {
        request = extend({}, option, { httpRequest: response });
        if (response.status >= 200 && response.status <= 299) {
          return response.json();
        }
      }).then((data) => {
        option.onSuccess(data, request);
      }).catch((error) => {
        option.onFailure(request);
      });
    },
  }),
},

```

```

}).executeQuery(new Query().take(8)).then((e) => {
//e.result will contain the records
});
`

```

Since the `CustomDataAdaptor` is extended from the `UrlAdaptor`, it expects response as a JSON object with properties `result` and `count` which contains the collection of entities and the total number of records respectively.

The sample response object should be as follows,

```

{
"result": [{..}, {..}, {..}, ...],
"count": 67
}
`

```

Performing CRUD action with CustomDataAdaptor

You can perform the CRUD actions using the `addRecord`, `updateRecord`, `deleteRecord` and `batchUpdate` methods.

```

`ts
import { DataManager, Query, CustomDataAdaptor, FetchOption } from '@syncfusion/ej2-data';
let createRequest: Function = (url: string, option: FetchOption) => {
let request: object;
fetch(SERVICE_URI, {
method: 'POST',
headers: {
'Content-Type': 'application/json; charset=utf-8',
},
}).then((response) => {
request = extend({}, option, { httpRequest: response });
if (response.status >= 200 && response.status <= 299) {
return response.json();
}
}).then((data) => {
option.onSuccess(data, request);
}).catch((error) => {

```

```

option.onFailure(request);
});
}
let baseUrl: string = "http://localhost:65327/Home/";
new DataManager({
  adaptor: new CustomDataAdaptor({
    getData: function (option: FetchOption) {
      createRequest(baseUrl + 'UrlDatasource', option);
    },
    addRecord: function (option: FetchOption) {
      createRequest(baseUrl + 'Insert', option);
    },
    updateRecord: function (option: FetchOption) {
      createRequest(baseUrl + 'Update', option);
    },
    deleteRecord: function (option: FetchOption) {
      createRequest(baseUrl + 'Delete', option);
    }
  })
  // to handle Batch operation
  //batchUpdate: function (option: FetchOption) {
  // createRequest(baseUrl + 'Delete', option);
  //}
})
}).executeQuery(new Query().take(8)).then((e) => {
  //e.result will contain the records
});
`

```

GraphQL Adaptor

The **GraphQLAdaptor** provides an option to retrieve data from the GraphQL server. It performs CRUD and data operations such as paging, sorting, filtering etc by sending the required arguments to the server.

You can provide the GraphQL query string by using the **query** property of the **GraphQLAdaptor**. Since, the **GraphQLAdaptor** is extended from the **UrlAdaptor**, it expects response as a JSON object with properties **result** and **count** which contains the collection of entities and the total number of records

respectively. The GraphQL response should be returned in JSON format like { "data": { ... } } with query name as field, you need to set the `result` and `count` properties to map the response.

`ts

```
import { DataManager, Query, GraphQLAdaptor } from '@syncfusion/ej2-data';
```

```
const SERVICE_URI: string = 'http://controller.com/actions';
```

```
new DataManager({
```

```
url: SERVICE_URI, adaptor: new GraphQLAdaptor({
```

```
response: {
```

```
result: 'getOrders.OrderData',
```

```
count: 'getOrders.OrderCount'
```

```
},
```

```
query: `query getOrders($datamanager: String) {
```

```
getOrders(datamanager: $datamanager) {
```

```
OrderCount,
```

```
OrderData{OrderID, CustomerID, EmployeeID, ShipCity, ShipCountry}
```

```
}
```

```
`
```

```
})
```

```
}).executeQuery(new Query().take(8)).then((e) => {
```

```
//e.result will contain the records
```

```
});
```

```
,
```

The Schema for the GraphQL server is

`ts

```
input OrderInput {
```

```
OrderID: Int!
```

```
CustomerID: String!
```

```
EmployeeID: Int!
```

```
ShipCity: String!
```

```
ShipCountry: String!
```

```
}
```

```
type Order {
```

```
OrderID: Int!
```

```

CustomerID: String!
EmployeeID: Int!
ShipCity: String!
ShipCountry: String!
}
type ReturnType {
  getOrders: [Order]
  count: Int
}
type Query {
  getOrders(datamanager: String): ReturnType
}
type Mutation {
  createOrder(value: OrderInput): Order!
  updateOrder(key: Int!, keyColumn: String, value: OrderInput): Order
  deleteOrder(key: Int!, keyColumn: String, value: OrderInput): Order!
}
`

```

The resolver for the corresponding action is

```

`ts
import { data } from "./db";
const resolvers = {
  Query: {
    getOrders: (parent, { datamanager }, context, info) => {
      if (datamanager.search) {
        // Perform searching
      }
      if (datamanager.sorted) {
        // Perform sorting
      }
      if (datamanager.where) {
        // Perform filtering
      }
    }
  }
}

```

```

if (datamanager.search) {
// Perform search
}
if (datamanager.skip && datamanager.take) {
// Perform Paging
}
return { OrderData: data, OrderCount: data.length };
},
Mutation: {
createOrder: (parent, { value }, context, info) => {
// Perform Insert
return value;
},
updateOrder: (parent, { key, keyColumn, value }, context, info) => {
// Perform Update
return value;
},
deleteOrder: (parent, { key, keyColumn, value }, context, info) => {
// Perform Delete
return value;
},
};
export default resolvers;
`

```

The query parameters will be send in a string format which contains the below details.

Parameters	Description
RequiresCounts	If it is true then the total count of records will be included in response.
Skip	Holds the number of records to skip.
Take	Holds the number of records to take.
Sorted	Contains details about current sorted column and its direction.

| **Where** | Contains details about current filter column name and its constraints. |

| **Group** | Contains details about current Grouped column names. |

Performing CRUD action with GraphQLAdaptor

You can perform the CRUD actions by returning the mutation queries inside the **getMutation** method based on the action.

```
`ts
import { DataManager, Query, GraphQLAdaptor } from '@syncfusion/ej2-data';
const SERVICE_URI: string = 'http://controller.com/actions';
new DataManager({
url: SERVICE_URI, adaptor: new GraphQLAdaptor({
response: {
result: 'getOrders.getOrders',
count: 'getOrders.count'
},
query: `query getOrders($datamanager: String) {
getOrders(datamanager: $datamanager) {
count,
getOrders{OrderID, CustomerID, EmployeeID, ShipCity, ShipCountry}
}
}`,
getMutation: function (action): string {
if (action === 'insert') {
return `mutation CreateOrderMutation($value: OrderInput!){
createOrder(value: $value){
OrderID, CustomerID, EmployeeID, ShipCity, ShipCountry
}}`;
}
if (action === 'update') {
return `mutation Update($key: ID!, $keyColumn: String,$value: OrderInput){
updateOrder(key: $key, keyColumn: $keyColumn, value: $value) {
OrderID, CustomerID, EmployeeID, ShipCity, ShipCountry
}
}`;
}
} else {
```

```

return `mutation Remove($key: ID!, $keyColumn: String, $value: OrderInput){
deleteOrder(key: $key, keyColumn: $keyColumn, value: $value) {
  OrderID, CustomerID, EmployeeID, ShipCity, ShipCountry
}
};
}
}
})
}.executeQuery(new Query().take(8)).then((e) => {
//e.result will contain the records
});
`

```

Writing custom adaptor

Sometimes the built-in adaptors does not meet your requirement. In such cases you can create your own adaptor.

To create and use custom adaptor, please refer to the below steps.

- Select an built-in adaptor which will act as base class for your custom adaptor.
- Override the desired method to achieve your requirement.
- Assign the custom adaptor to the `adaptor` property of **DataManager**.

For the sake of demonstrating custom adaptor approach, we are going to see how to add serial number for the records by overriding the built-in response processing using `processResponse` method of the `ODataV4Adaptor`.

INDEX.JS

```

var template =
`<tr><td>${Sno}</td><td>${CustomerID}</td><td>${EmployeeID}</td></tr>`;

var compiledFunction = ej.base.compile(template);

class SerialNoAdaptor extends ej.data.ODataV4Adaptor {
  processResponse() {
    let i = 0;
    // calling the base class processResponse function
    let original = super.processResponse.apply(this, arguments);
    // Adding serial number
    original.forEach((item) => item['Sno'] = ++i);
    return original;
  }
}

const SERVICE_URI =
'https://services.odata.org/V4/Northwind/Northwind.svc/Orders/';

```

```

var table = document.getElementById('datatable');

new ej.data.DataManager({ url: SERVICE_URI, adaptor: new SerialNoAdaptor()
})
    .executeQuery(new ej.data.Query().take(8))
    .then((e) => {
        (e.result).forEach((data) => {
            table.appendChild(compiledFunction(data)[0]);
        });
    });

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Grid"></div>
    <table id="datatable" class="e-table">
      <thead>
        <tr><th>Order ID</th><th>Customer ID</th><th>Employee
ID</th></tr>
      </thead>

```

```

        <tbody>
        </tbody>
    </table>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Querying in ##Platform_Name## Data control

In this section, you will see in detail about how to build query using [Query](#) class and consume the data source.

Specifying resource name using `from`

The [from](#) method is used to specify the resource name or table name from where the data should be retrieved.

INDEX.JS

```

var template =
'<tr><td>${OrderID}</td><td>${CustomerID}</td><td>${EmployeeID}</td></tr>';
var compiledFunction = ej.base.compile(template);
const SERVICE_URI =
'https://services.odata.org/V4/Northwind/Northwind.svc/';
var table = (document.getElementById('datatable'));
new ej.data.DataManager({ url: SERVICE_URI, adaptor: new
ej.data.ODataV4Adaptor })
    .executeQuery(new ej.data.Query().from('Orders').take(8)).then((e) => {
        (e.result).forEach((data) => {
            table.appendChild(compiledFunction(data)[0]);
        });
    });

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
        <table id="datatable" class="e-table">
            <thead>
                <tr><th>Order ID</th><th>Customer ID</th><th>Employee
ID</th></tr>
            </thead>
            <tbody>
            </tbody>
        </table>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Projection using `select`

The [select](#) method is used to select particular fields or columns from the data source.

INDEX.JS

```

var template =
'<tr><td>${OrderID}</td><td>${CustomerID}</td><td>${EmployeeID}</td></tr>';
var compiledFunction = ej.base.compile(template);
const SERVICE_URI =
'https://services.odata.org/V4/Northwind/Northwind.svc/Orders/';
var table = (document.getElementById('datatable'));
new ej.data.DataManager({ url: SERVICE_URI, adaptor: new
ej.data.ODataV4Adaptor })
    .executeQuery(new ej.data.Query().select(['OrderID', 'CustomerID',
'EmployeeID']).take(8))

```

```
.then((e) => {
    (e.result).forEach((data) => {
        table.appendChild(compiledFunction(data)[0]);
    });
});
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Grid"></div>
    <table id="datatable" class="e-table">
      <thead>
        <tr><th>Order ID</th><th>Customer ID</th><th>Employee
ID</th></tr>
      </thead>
      <tbody>
      </tbody>
    </table>
  </div>
</script>
```

```

var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Eager loading navigation properties

You can use the [expand](#) method to eagerly load navigation properties. The navigation properties values are accessed using appropriate field names separated by dot(.) sign.

INDEX.JS

```

var template =
'<tr><td>${OrderID}</td><td>${CustomerID}</td><td>${Employee.FirstName}</td></tr>';
var compiledFunction = ej.base.compile(template);
const SERVICE_URI =
'https://services.odata.org/V4/Northwind/Northwind.svc/Orders/';
var table = (document.getElementById('datatable'));
table.innerHTML = '<tr><th>OrderID</th><th>CustomerID</th><th>Employee
Name</th></tr>';
new ej.data.DataManager({ url: SERVICE_URI, adaptor: new
ej.data.ODataV4Adaptor })
    .executeQuery(new ej.data.Query().expand('Employee').select(['OrderID',
'CustomerID', 'Employee.FirstName']).take(8))
    .then((e) => {
        (e.result).forEach((data) => {
            table.appendChild(compiledFunction(data)[0]);
        });
    });

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
<title>EJ2 Grid</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
        <table id="datatable" class="e-table">
            <thead>
                <tr><th>Order ID</th><th>Customer ID</th><th>Employee
ID</th></tr>
            </thead>
            <tbody>
            </tbody>
        </table>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Sorting

You can use the [sortBy](#) method to perform sort operation in the data source. Default sorting order is **ascending**. To change the sort order, either you can specify the second argument of [sortBy](#) as **descending** or use the [sortByDesc](#) method.

INDEX.JS

```

var template =
'<tr><td>${OrderID}</td><td>${CustomerID}</td><td>${EmployeeID}</td></tr>';
var compiledFunction = ej.base.compile(template);
const SERVICE_URI =
'https://services.odata.org/V4/Northwind/Northwind.svc/Orders/';
var table = (document.getElementById('datatable'));
new ej.data.DataManager({ url: SERVICE_URI, adaptor: new
ej.data.ODataV4Adaptor })
    .executeQuery(new ej.data.Query().sortBy('CustomerID',
'descending').take(8))
    .then((e) => {
        (e.result).forEach((data) => {

```



```

        table.appendChild(compiledFunction(data)[0]);
    });
});

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Grid"></div>
    <table id="datatable" class="e-table">
      <thead>
        <tr><th>Order ID</th><th>Customer ID</th><th>Employee
ID</th></tr>
      </thead>
      <tbody>
      </tbody>
    </table>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {

```

```

    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Multi sorting can be performed by simply chaining the multiple `sortBy` methods.

Filtering

You can use the [where](#) method to build filter criteria which allows you to get reduced view of records. The [where](#) method can also be chained to form multiple filter criteria.

INDEX.JS

```

var template =
'<tr><td>${OrderID}</td><td>${CustomerID}</td><td>${EmployeeID}</td></tr>';
var compiledFunction = ej.base.compile(template);
const SERVICE_URI =
'https://services.odata.org/V4/Northwind/Northwind.svc/Orders/';
let table = (document.getElementById('datatable'));
new ej.data.DataManager({ url: SERVICE_URI, adaptor: new
ej.data.ODataV4Adaptor })
    .executeQuery(new ej.data.Query().where('EmployeeID', 'equal',
3).take(8))
    .then((e) => {
        (e.result).forEach((data) => {
            table.appendChild(compiledFunction(data)[0]);
        });
    });

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
        <table id="datatable" class="e-table">
            <thead>
                <tr><th>Order ID</th><th>Customer ID</th><th>Employee
ID</th></tr>
            </thead>
            <tbody>
            </tbody>
        </table>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Filter Operators

Filter operators are generally used to specify the filter type. The various filter operators supported by **DataManager** is listed below.

- greaterthan
- greaterthanorequal
- lessthan
- lessthanorequal
- equal
- notequal
- startswith
- endswith
- contains

These filter operators are used for creating filter query using [where](#) method and [Predicate](#) class.

Build complex filter criteria using `Predicate`

Sometimes chaining [where](#) method is not sufficient to create very complex filter criteria, in such cases we can use [Predicate](#) class to create composite filter criteria.

INDEX.JS

```

var template =
'<tr><td>${OrderID}</td><td>${CustomerID}</td><td>${EmployeeID}</td></tr>';
var compiledFunction = ej.base.compile(template);
const SERVICE_URI =
'https://services.odata.org/V4/Northwind/Northwind.svc/Orders/';
let table = (document.getElementById('datatable'));
//Building complex filter criteria using `Predicate`
var predicate = new ej.data.Predicate('EmployeeID', 'equal', 3);
predicate = predicate.or('EmployeeID', 'equal', 2);
new ej.data.DataManager({ url: SERVICE_URI, adaptor: new
ej.data.ODataV4Adaptor })
    .executeQuery(new ej.data.Query().where(predicate).take(8))
    .then((e) => {
        (e.result).forEach((data) => {
            table.appendChild(compiledFunction(data)[0]);
        });
    });

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>

```

```

<body>

  <div id="container">
    <div id="Grid"></div>
    <table id="datatable" class="e-table">
      <thead>
        <tr><th>Order ID</th><th>Customer ID</th><th>Employee
ID</th></tr>
      </thead>
      <tbody>
      </tbody>
    </table>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Searching

You can use the [search](#) method to create search criteria, it differs from the filter in the way that search criteria will applied to all fields in the data source whereas filter criteria will be applied to a particular field.

INDEX.JS

```

var template =
'<tr><td>${OrderID}</td><td>${CustomerID}</td><td>${EmployeeID}</td></tr>';
var compiledFunction = ej.base.compile(template);
var table = (document.getElementById('datatable'));
new ej.data.DataManager(data)
  .executeQuery(new ej.data.Query().search('VI', ['CustomerID']))
  .then((e) => {
    (e.result).forEach((data) => {
      table.appendChild(compiledFunction(data)[0]);
    });
  });

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
        <table id="datatable" class="e-table">
            <thead>
                <tr><th>Order ID</th><th>Customer ID</th><th>Employee
ID</th></tr>
            </thead>
            <tbody>
            </tbody>
        </table>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

You can search particular fields by passing the field name collection in the second argument of [search](#) method.

Grouping

DataManager allow you to group records by category. The [group](#) method is used to add group query.

INDEX.JS

```

var template = '<tr><td>${field} -
${key}</td><td></td><td></td></tr>${for(item of

```

```

items) }<tr><td>${item.OrderID}</td><td>${item.CustomerID}</td><td>${item.Emp
loyeeID}</td></tr>${/for}';
var compiledFunction = ej.base.compile(template);
const SERVICE_URI =
'https://services.odata.org/V4/Northwind/Northwind.svc/Orders/';
var table = (document.getElementById('datatable'));
new ej.data.DataManager({ url: SERVICE_URI, adaptor: new
ej.data.ODataV4Adaptor })
    .executeQuery(new ej.data.Query().group('CustomerID').take(8))
    .then((e) => {
        (e.result).forEach((data) => {
            table.appendChild(compiledFunction(data)[0]);
        });
    });
});

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Grid"></div>
    <table id="datatable" class="e-table">

```

```

        <thead>
            <tr><th>Order ID</th><th>Customer ID</th><th>Employee
ID</th></tr>
        </thead>
        <tbody>
        </tbody>
    </table>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Multiple grouping can be done by simply chaining the [group](#) method.

Paging

You can query paged data using [page](#) method. This allow you to query particular set of records based on the page size and index.

INDEX.JS

```

var template =
'<tr><td>${OrderID}</td><td>${CustomerID}</td><td>${EmployeeID}</td></tr>';
var compiledFunction = ej.base.compile(template);
const SERVICE_URI =
'https://services.odata.org/V4/Northwind/Northwind.svc/Orders/';
var table = (document.getElementById('datatable'));
new ej.data.DataManager({ url: SERVICE_URI, adaptor: new
ej.data.ODataV4Adaptor })
    .executeQuery(new ej.data.Query().page(2, 8))
    .then((e) => {
        (e.result).forEach((data) => {
            table.appendChild(compiledFunction(data)[0]);
        });
    });

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
        <table id="datatable" class="e-table">
            <thead>
                <tr><th>Order ID</th><th>Customer ID</th><th>Employee
ID</th></tr>
            </thead>
            <tbody>
            </tbody>
        </table>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Aggregation

The [aggregate](#) method allows you to get aggregated value for a field based on the type. The built-in aggregate types are,

- sum
- average
- min
- max
- count
- truecount
- falsecount

INDEX.JS

```

var template =
'<tr><td>${OrderID}</td><td>${CustomerID}</td><td>${EmployeeID}</td></tr>';
var compiledFunction = ej.base.compile(template);
var footerFn = ej.base.compile('<tr><td></td><td></td><td>Minimum:
${min}</td></tr>');
const SERVICE_URI =
'https://services.odata.org/V4/Northwind/Northwind.svc/Orders/';
var table = (document.getElementById('datatable'));
new ej.data.DataManager({ url: SERVICE_URI, adaptor: new
ej.data.ODataV4Adaptor })
    .executeQuery(new
ej.data.Query().take(5).requiresCount().aggregate('min', 'EmployeeID'))
    .then((e) => {
        (e.result).forEach((data) => {
            table.appendChild(compiledFunction(data)[0]);
        });
        table.appendChild(footerFn({ min: e.aggregates['EmployeeID - min']
    })[0]);
    });

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>

```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
        <table id="datatable" class="e-table">
            <thead>
                <tr><th>Order ID</th><th>Customer ID</th><th>Employee
ID</th></tr>
            </thead>
            <tbody>
            </tbody>
        </table>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Hierarchical query

You can use the [hierarchy](#) method to build nested query. The hierarchical queries are commonly required when you use foreign key binding.

The [foreignKey](#) method is used to specify the key field of the foreign table and the second argument of the [hierarchy](#) method accepts a selector function which selects the records from the foreign table.

INDEX.JS

```

var template =
'<tr><td>${OrderID}</td><td>${CustomerID}</td><td>${EmployeeID}</td></tr>'
var group = '<tr><td colspan=3>' +
'<table id="datatable" class="e-
table"><tr><th>ID</th><th>Price</th><th>Quantity</th></tr>' +
'${for(detail of
Order_Details)}<tr><td>${detail.ProductID}</td><td>${detail.UnitPrice}</td><
td>${detail.Quantity}</td></tr>${/for}' +
'<table></td></tr>';
var compiledFunction = ej.base.compile(template);
var groupFn = ej.base.compile(group);
const SERVICE_URI =
'https://services.odata.org/V4/Northwind/Northwind.svc/';
var table = (document.getElementById('datatable'));
new ej.data.DataManager({ url: SERVICE_URI, adaptor: new
ej.data.ODataV4Adaptor })
    .executeQuery(new ej.data.Query().from('Orders').take(3).hierarchy(
        new ej.data.Query()
            .foreignKey("OrderID")
            .from("Order_Details")
            .sortBy("Quantity"),
        function () {

```

```

        // Selective loading of child elements
        return [10248, 10249, 10250]
    }
    })
    .then((e) => {
        (e.result).forEach((data) => {
            table.appendChild(compiledFunction(data) [0]);
            table.appendChild(groupFn(data) [0]);
        });
    });
});

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Grid"></div>
    <table id="datatable" class="e-table">
      <thead>
        <tr><th>Order ID</th><th>Customer ID</th><th>Employee
ID</th></tr>

```

```

        </thead>
        <tbody>
        </tbody>
    </table>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Manipulation in ##Platform_Name## Data control

In this section, you will see in detail about how to manipulate data using **DataManager**. The **DataManager** can create, update and delete records either in local data source or remote data source.

Each data sources uses different way in handling the CRUD operations and hence **DataManager** uses data adaptors to manipulate data that can be understood by a particular data source.

Insert

The [insert](#) method of **DataManager** is used to add new record to the data source. For remote data source, the new record will be send along with the request to the server.

INDEX.JS

```

var template =
'<tr><td>${OrderID}</td><td>${CustomerID}</td><td>${EmployeeID}</td></tr>';
var compiledFunction = ej.base.compile(template);
var table = (document.getElementById('datatable'));
var dm = new ej.data.DataManager(data.slice(0, 4));
dm.executeQuery(new ej.data.Query())
    .then((e) => {
        (e.result).forEach((data) => {

table.tBodies[0].appendChild(compiledFunction(data)[0].firstChild);
        });
    });
var button = document.getElementById('manipulate');
var orderid = document.getElementById('OrderID');
var cusid = document.getElementById('CustomerID');
var empid =document.getElementById('EmployeeID');
button.onclick = function() {
    var data = {
        OrderID: orderid.value,
        CustomerID: cusid.value,
        EmployeeID: empid.value
    };
    if (!data.OrderID) { return; }
    dm.insert(data);
    dm.executeQuery(new ej.data.Query())
        .then((e) => {
            table.tBodies[0].innerHTML = '';
            (e.result).forEach((data) => {

```

```

table.tBodies[0].appendChild(compiledFunction(data)[0].firstChild);
    });
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <style>
    .e-form {
      display: block;
      padding-bottom: 10px;
    }
    .e-form input {
      width: 15%;
    }
  </style>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">

```

```

        <div class="e-form">
            <input type="number" id="OrderID" placeholder="Order ID">
            <input type="text" id="CustomerID" placeholder="Customer ID">
            <input type="number" id="EmployeeID" placeholder="Employee ID">
            <input type="button" value="Insert" id="manipulate">
        </div>
        <table id="datatable" class="e-table">
            <thead>
                <tr>
                    <th>Order ID</th>
                    <th>Customer ID</th>
                    <th>Employee ID</th>
                </tr>
            </thead>
            <tbody>
            </tbody>
        </table>
    </div>
    <script>
    var ele = document.getElementById('container');
    if(ele) {
        ele.style.visibility = "visible";
    }
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

In remote data sources, when the primary key field is an identity field, then it is advised to return the created data in the response.

Update

The [update](#) method of **DataManager** is used to modify/update a record in the data source. For remote data source, the modified record will be send along with the request to the server.

INDEX.JS

```

var template =
'<tr><td>${OrderID}</td><td>${CustomerID}</td><td>${EmployeeID}</td></tr>';
var compiledFunction = ej.base.compile(template);
var table = (document.getElementById('datatable'));
var dm = new ej.data.DataManager(data.slice(0, 4));
dm.executeQuery(new ej.data.Query()
    .then((e) => {
        (e.result).forEach((data) => {
            table.tBodies[0].appendChild(compiledFunction(data)[0].firstChild);
        });
    }));
var button = document.getElementById('manipulate');
var orderid = document.getElementById('OrderID');
var cusid = document.getElementById('CustomerID');
var empid = document.getElementById('EmployeeID');
button.onclick = () => {
    var data = {
        OrderID: +orderid.value,
        CustomerID: cusid.value,

```

```

        EmployeeID: +empid.value
    };
    if (!data.OrderID) { return; }
    dm.update('OrderID', data);
    dm.executeQuery(new ej.data.Query())
        .then((e) => {
            table.tBodies[0].innerHTML = '';
            (e.result).forEach((data) => {

table.tBodies[0].appendChild(compiledFunction(data)[0].firstChild);
            });
        });
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <style>
    .e-form {
      display: block;
      padding-bottom: 10px;
    }
    .e-form input {
      width: 15%;
    }
  </style>

```



```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="e-form">
            <input type="number" id="OrderID" placeholder="Order ID">
            <input type="text" id="CustomerID" placeholder="Customer ID">
            <input type="number" id="EmployeeID" placeholder="Employee ID">
            <input type="button" value="Insert" id="manipulate">
        </div>
        <table id="datatable" class="e-table">
            <thead>
                <tr>
                    <th>Order ID</th>
                    <th>Customer ID</th>
                    <th>Employee ID</th>
                </tr>
            </thead>
            <tbody>
            </tbody>
        </table>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Primary key name is required by the [update](#) method to find the record to be updated.

Remove

The [remove](#) method of **DataManager** is used to remove a record from the data source. For remote data source, the record details such as primary key and data will be send along with the request to the server.

INDEX.JS

```

var template =
'<tr><td>${OrderID}</td><td>${CustomerID}</td><td>${EmployeeID}</td></tr>';
var compiledFunction = ej.base.compile(template);
var table = (document.getElementById('datatable'));
var dm = new ej.data.DataManager(data.slice(0, 4));
dm.executeQuery(new ej.data.Query())
    .then((e) => {
        (e.result).forEach((data) => {
            table.tBodies[0].appendChild(compiledFunction(data)[0].firstChild);
        });
    });

```

```

var button = document.getElementById('manipulate');
button.value = 'Remove';
var orderid = document.getElementById('OrderID');
document.getElementById('CustomerID').style.display = 'none';
document.getElementById('EmployeeID').style.display = 'none';
button.onclick = () => {
    if (!orderid.value) { return; }
    dm.remove('OrderID', { OrderID: +orderid.value });
    dm.executeQuery(new ej.data.Query())
        .then((e) => {
            table.tBodies[0].innerHTML = '';
            (e.result).forEach((data) => {

table.tBodies[0].appendChild(compiledFunction(data)[0].firstChild);
            });
        });
};
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <style>
        .e-form {
            display: block;
            padding-bottom: 10px;
        }
        .e-form input {

```

```

        width: 15%;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="e-form">
            <input type="number" id="OrderID" placeholder="Order ID">
            <input type="text" id="CustomerID" placeholder="Customer ID">
            <input type="number" id="EmployeeID" placeholder="Employee ID">
            <input type="button" value="Insert" id="manipulate">
        </div>
        <table id="datatable" class="e-table">
            <thead>
                <tr>
                    <th>Order ID</th>
                    <th>Customer ID</th>
                    <th>Employee ID</th>
                </tr>
            </thead>
            <tbody>
            </tbody>
        </table>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Primary key name and its value are required to find the record to be removed.

Batch Edit Operation

DataManager supports batch processing for the CRUD operations. You can use the [saveChanges](#) method to batch the edit operation. For remote data source, requests to add, remove and change are handled altogether at a time rather than passing the request separately for each operation.

INDEX.JS

```

var template =
'<tr><td>${OrderID}</td><td>${CustomerID}</td><td>${EmployeeID}</td></tr>';
var compiledFunction = ej.base.compile(template);
var table = (document.getElementById('datatable'));
var dm = new ej.data.DataManager({ json: (data).slice(0, 4) });
dm.executeQuery(new ej.data.Query())
    .then((e) => {
        (e.result).forEach((data) => {

```

```

table.tBodies[0].appendChild(compiledFunction(data)[0].firstChild);
    });
});
var changes = {
    changedRecords: [], addedRecords: [], deletedRecords: []
};
var orderid = document.getElementById('OrderID');
var cusid = document.getElementById('CustomerID');
var empid = document.getElementById('EmployeeID');
document.getElementById('added').onclick = () => {
    changes.addedRecords.push({
        OrderID: +orderid.value,
        CustomerID: cusid.value,
        EmployeeID: +empid.value
    });
    orderid.value = cusid.value = empid.value = null;
};
document.getElementById('changed').onclick = () => {
    changes.changedRecords.push({
        OrderID: +orderid.value,
        CustomerID: cusid.value,
        EmployeeID: +empid.value
    });
    orderid.value = cusid.value = empid.value = null;
};
document.getElementById('deleted').onclick = () => {
    changes.deletedRecords.push({
        OrderID: +orderid.value,
        CustomerID: cusid.value,
        EmployeeID: +empid.value
    });
    orderid.value = cusid.value = empid.value = null;
};
document.getElementById('save').onclick = () => {
    dm.saveChanges(changes, 'OrderID');
    changes = { changedRecords: [], addedRecords: [], deletedRecords: [] };
    dm.executeQuery(new ej.data.Query())
        .then((e) => {
            table.tBodies[0].innerHTML = '';
            (e.result).forEach((data) => {

table.tBodies[0].appendChild(compiledFunction(data)[0].firstChild);
            });
        });
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
.e-form {
    display: block;
    padding-bottom: 10px;
}
.e-form input {
    width: 15%;
}
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

<div id="container">
<div class="e-form">
    <input type="number" id="OrderID" placeholder="Order ID">
    <input type="text" id="CustomerID" placeholder="Customer ID">
    <input type="number" id="EmployeeID" placeholder="Employee ID">
    <input type="button" value="Insert" id="added">
    <input type="button" value="Update" id="changed">
    <input type="button" value="Remove" id="deleted">
</div>
<div class="e-form">
    <label>Click to Save changes:</label>
    <input type="button" value="Save Changes" id="save"
style="width: 20%">
</div>
<table id="datatable" class="e-table">
    <thead>
        <tr>
            <th>Order ID</th>

```

```

                <th>Customer ID</th>
                <th>Employee ID</th>
            </tr>
        </thead>
        <tbody>
        </tbody>
    </table>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

State persistence in ##Platform_Name## Data control

State persistence refers to the ability of the DataManager to maintain its state in the browser's localStorage, even when the browser is refreshed or when navigating to a different page within the same browser. To enable this feature, you need to set the `id` and `enablePersistence` properties in the DataManager. This allows the DataManager's query object to persistently store in the local storage.

`ts

```
import { DataManager, Query, UrlAdaptor } from "@syncfusion/ej2-data";
```

```
let SERVICE_URI =
```

```
"https://services.syncfusion.com/js/production/api/UrlDataSource";
```

```
new DataManager({
```

```
url: SERVICE_URI,
```

```
adaptor: new UrlAdaptor(),
```

```
//Mandatory properties to use state persistence.
```

```
enablePersistence: true,
```

```
id: "johnDoe",
```

```
})
```

```
.executeQuery(new Query().take(8))
```

```
.then((e) => {
```

```
//e.result will contain the records
```

```
});
```

```
`
```

Preventing a query from persistence

By default, the DataManager can persist various types of queries, such as sorting, searching, filtering, and selection queries. However, there may be cases where you want to exclude specific queries from

persistence. To achieve this, you can utilize the `ignoreOnPersist` property and specify the queries you wish to exclude. Refer to the table below for the naming conventions of DataManager queries:

Sorting: `onSortBy`

Searching: `onSearch`

Selection: `onSelect`

Filtering: `onWhere`

Grouping: `onGroup`

The `ignoreOnPersist` property type is an array, allowing you to exclude multiple queries from persistence according to your requirements.

Refer to the following example, which demonstrates how to exclude the sorting query ("`onSortBy`") and the search query ("`onSearch`") from being persisted in the DataManager:

```
`ts
import { DataManager, Query, UrlAdaptor } from "@syncfusion/ej2-data";
let SERVICE_URI =
"https://services.syncfusion.com/js/production/api/UrlDataSource";
new DataManager({
url: SERVICE_URI,
adaptor: new UrlAdaptor(),
enablePersistence: true,
id: "DataManagerid",
//sort and search query won't persist now.
ignoreOnPersist: ["onSortBy", "onSearch"],
})
.executeQuery(new Query().sortBy("Designation", "descending").take(8))
.then((e) => {
//e.result will contain the records
});
`
```

How to get or set the existing persisted data

To access or modify the existing persisted data in the DataManager, you can utilize the [getPersistedData](#) and [setPersistData](#) methods available in the DataManager.

The [getPersistedData](#) method allows you to retrieve the existing persisted data. It takes a single argument, which is the DataManager's `id`. By passing the DataManager's id, you can retrieve the persisted data associated with the DataManager from the `window.localStorage`. Here is an example of how to use the `getPersistedData` method:

```
`ts
import { DataManager, Query, UrlAdaptor } from "@syncfusion/ej2-data";
let SERVICE_URI =
"https://services.syncfusion.com/js/production/api/UrlDataSource";
let dataManager = new DataManager({
url: SERVICE_URI,
adaptor: new UrlAdaptor(),
enablePersistence: true,
id: "Johndoe",
});
let persistedQuery = dataManager.getPersistedData("Johndoe");
`
```

On the other hand, the [setPersistData](#) method enables you to add a query to the existing persisted data. It accepts three arguments:

Original event: Set this argument to null.

Id: Pass the DataManager's id value.

Query: Provide the query that you want to add to the persisted data.

Here is an example demonstrating how to add a query to the existing persisted data:

```
`ts
import { DataManager, Query, UrlAdaptor } from "@syncfusion/ej2-data";
let SERVICE_URI =
"https://services.syncfusion.com/js/production/api/UrlDataSource";
let dataManager = new DataManager({
url: SERVICE_URI,
adaptor: new UrlAdaptor(),
enablePersistence: true,
id: "Johndoe",
});
let query = new Query().sortBy("Designation", "descending").take(8);
dataManager.setPersistData(null, "Johndoe", query);
`
```

By using the setPersistData method, you can append the specified query to the DataManager's existing persisted data.

Restoring the initial state of Datamanager

If you have enabled the `enablePersistence` feature in the DataManager, it automatically retains the previous state when the browser is refreshed or reloaded. However, there might be situations where you want to clear the persistence and load the initial state of the DataManager. In such cases, you can utilize the `clearPersistence()` method provided by the DataManager.

Here is a code example demonstrating how to clear the persistence in the DataManager:

```
`ts
import { DataManager, Query, UrlAdaptor } from "@syncfusion/ej2-data";
let SERVICE_URI =
"https://services.syncfusion.com/js/production/api/UrlDataSource";
let dataManager = new DataManager({
url: SERVICE_URI,
adaptor: new UrlAdaptor(),
enablePersistence: true,
id: "dataManagerid",
});
let query = new Query().sortBy("Designation", "descending").take(8);
//sets persist query to browser storage.
dataManager.setPersistData(null, "Johndoe", query);
//clears the persisted query.
dataManager.clearPersistence();
`
```

By invoking the `clearPersistence()` method, you can remove the persisted data and restore the DataManager to its initial state.

Use case example demonstrating state persistence with the DataManager

This demonstration involves two controls, namely the **Grid** control and the **Chart** control, which both fetch data from the same instance of the DataManager, which has the state persistence feature enabled.

The Grid control is responsible for displaying the entire dataset, while the Chart control presents user reviews based on specific data from the "column name" field. Both controls are associated with the same DataManager instance and it has enabled the state persistence feature. The query state of the DataManager is automatically saved in the browser's local storage as the user applies filtering and sorting actions. In both controls are reloaded the data with the last persisted state while refreshing or reloading the browser.

In this demo, the filter query and sort query are persisted, whereas the search query is not persisted. The `onSearch` query is excluded from persistence by setting it in the `ignoreOnPersist` property of the DataManager.

For a more detailed explanation and steps of this use case, refer to the following:

Step 1: To initiate the demo, users are required to select a username from the dropdown list. After making a selection, the Grid and Chart controls will load with initial data using the DataManager. Specifically for this demo, the DataManager's id will be set to the chosen username. The DataManager will then store the query with this id in the window.localStorage. Refer to the code example for your reference:

Step 2: This demo allows you to select Grid items by clicking checkboxes and adding them to your cart using the "Add" button in the toolbar. Additionally, you can sort the products from high price to low price by clicking the "Price Low-High" and "Price High-Low" buttons. Furthermore, you can view the added products from the wishlist by clicking the wishlist icon. All this information is persisted and stored by the DataManager based on the user ID.

You also can filter the product items using the product category filter. However, this category filter is also not persisted.

The Chart control allows you to see the product reviews.

Step 3: To log out, simply use the "Logout" button.

Step 4: After logging out or refreshing the browser, you will need to select a username from the dropdown list once again (repeat step 1). However, the Grid data will now be loaded based on the last persisted wishlist, associated with the chosen username. The complete example is as follows:

To clear the wishlist for a specific user, click the "Clear Wishlist" button. This will remove all the saved wishlist items for that user.

[Here](#), you can find the Use case example demonstrating state persistence with the DataManager.

How to in ##Platform_Name## Data control

Work in offline mode

On remote data binding, every time invoking [executeQuery](#) will send request to the server and the query will be processed on server-side. To avoid post back to server on calling [executeQuery](#), you can set the **DataManager** to load all the data on initialization time and make the query processing in client-side. To enable this behavior, you can use **offline** property of **DataManager**.

INDEX.JS

```
var template =
'<tr><td>${OrderID}</td><td>${CustomerID}</td><td>${EmployeeID}</td></tr>';
var compiledFunction = ej.base.compile(template);
const SERVICE_URI =
'https://services.odata.org/V4/Northwind/Northwind.svc/Orders/?$top=7';
var table = (document.getElementById('datatable'));
var dm = new ej.data.DataManager({ url: SERVICE_URI, adaptor: new
ej.data.ODataV4Adaptor, offline: true });
dm.ready.then((e) => {
    (e.result).forEach((data) => {
        table.appendChild(compiledFunction(data)[0]);
    });
});
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
<title>EJ2 Grid</title>
```

```

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
        <table id="datatable" class="e-table">
            <thead>
                <tr><th>Order ID</th><th>Customer ID</th><th>Employee
ID</th></tr>
            </thead>
            <tbody>
            </tbody>
        </table>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The loaded data will be cached in the `json` property of **DataManager**.

Sending additional parameters to server

You can use the [addParams](#) method of [Query](#) class, to add custom parameter to the data request.

INDEX.JS

```
var template =
'<tr><td>${OrderID}</td><td>${CustomerID}</td><td>${EmployeeID}</td></tr>';
var compiledFunction = ej.base.compile(template);
const SERVICE_URI =
'https://services.odata.org/V4/Northwind/Northwind.svc/Orders';
var table = (document.getElementById('datatable'));
new ej.data.DataManager({ url: SERVICE_URI, adaptor: new
ej.data.ODataV4Adaptor })
    .executeQuery(new ej.data.Query().addParams('$top', '7'))
    .then((e) => {
        (e.result).forEach((data) => {
            table.appendChild(compiledFunction(data)[0]);
        });
    });
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
```

```

<body>

  <div id="container">
    <div id="Grid"></div>
    <table id="datatable" class="e-table">
      <thead>
        <tr><th>Order ID</th><th>Customer ID</th><th>Employee
ID</th></tr>
      </thead>
      <tbody>
      </tbody>
    </table>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Adding custom headers

You can add custom headers to the request made by **DataManager** using the **headers** property.

```

`ts
import { DataManager, Query, ReturnOption, ODataV4Adaptor } from '@syncfusion/ej2-data';
const SERVICE_URI: string = 'https://services.odata.org/V4/Northwind/Northwind.svc/Orders/';
new DataManager({ url: SERVICE_URI, adaptor: new ODataV4Adaptor, headers:[{ 'syncfusion': 'true' }] })
.executeQuery(new Query())
.then((e: ReturnOption) => {
//get result from e.result
});
`

```

Adding custom headers while making cross domain request will initiate preflight request.

DatePicker

Date range in ##Platform_Name## Datepicker control

DatePicker provides an option to select a date value within a specified range by using the [min](#) and [max](#) properties. Always the min value has to be lesser than the max value.

When the min and max properties are configured and the selected date value is out-of-range or invalid, then the model value will be set to **out of range** date value or **null** respectively with highlighted **error** class to indicates the date is out of range or invalid.

The value property depends on the min/max with respect to [strictMode](#) property.

The below example allows selecting a date within the range from 7th to 27th day in a month.

INDEX.JS

```
ej.base.enableRipple(true);

//Creates a DatePicker with min and max properties.
var today = new Date();
var currentYear = today.getFullYear();
var currentMonth = today.getMonth();
var currentDay = today.getDate();
var datepicker = new ej.calendars.DatePicker({
    //sets the min date
    min: new Date(currentYear, currentMonth, 7),
    //sets the max date
    max: new Date(currentYear, currentMonth, 27),
    //sets the value
    value: new Date(new Date().setDate(14))
});
datepicker.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DatePicker control</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-calendars/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <input id="element" type="text">
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
```

```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

If the value of `min` or `max` properties changed through code behind, then you have to update the `value` property to set within the range.

Date format in ##Platform_Name## DatePicker control

Date format is a way of representing the date value in different string format in the textbox.

By default, the DatePicker's format is based on the culture. You can also set the own custom format by using the [format](#) property.

Once the date format property has been defined it will be common to all the cultures.

To know more about the date format standards, refer to the [Internationalization Date Format](#) section.

The following example demonstrates the DatePicker with the custom format (`yyyy-MM-dd`).

INDEX.JS

```
var datepicker = new ej.calendars.DatePicker({
    placeholder: 'Choose a date',
    value: new Date(),
    // sets the format.
    format: 'yyyy-MM-dd'
});
datepicker.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DatePicker control</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-calendars/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>
```

```

    <div id="container">
        <input id="element" type="text">
    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Date masking in ##Platform_Name## Datepicker control

DatePicker has `enableMask` property that provides the option to enable the built-in date masking support. Also, you must inject the MaskedDateTime module to enable the masking support.

INDEX.JS

```

var datePickerObject = new ej.calendars.DatePicker({
    enableMask: true
});
datePickerObject.appendTo('#mask');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 DatePicker control</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="container">
        <input id="mask">

```



```

    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The mask pattern is defined based on the provided date format to the component. If the format is not specified, the mask pattern is formed based on the default format of the current culture.

| Keys | Actions |

| --- | --- |

| Up / Down arrows | To increment and decrement the selected portion of the date. |

| Left / Right arrows and Tab | To navigate the selection from one portion to next portion |

The following example demonstrates default and custom format of DatePicker component with mask.

INDEX.JS

```

var datePickerObject = new ej.calendars.DatePicker({
    enableMask: true
});
datePickerObject.appendTo('#mask');
var datePickerFormat = new ej.calendars.DatePicker({
    enableMask: true,
    format: "M/d/yyyy",
});
datePickerFormat.appendTo('#format');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 DatePicker control</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
  <body>

    <div id="container" style="margin:50px auto 0; width:250px;">
      <br><br>
      <label class="custom-input-label">Mask support with default
format</label>
      <br><br>
      <input id="mask">
      <br><br><br>
      <label class="custom-input-label">Mask support with custom
format</label>
      <br><br>
      <input id="format">
    </div>

<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Configure Mask Placeholder

You can change mask placeholder value through property `maskPlaceholder`. By default , it takes the full name of date and time co-ordinates such as `day`, `month`, `year`, `hour` etc.

While changing to a culture other than `English`, ensure that locale text for the concerned culture is loaded through load method of `L10n` class for mask placeholder values like below.

```
`ts
```

```
//Load the L10n from ej2-base
```

```
import { L10n } from '@syncfusion/ej2-base';
```

```
//load the locale object to set the localized mask placeholder value
```

```
L10n.load({
```

```
'de': {
```

```
'datepicker': { day: 'Tag' , month: 'Monat', year: 'Jahr' }
```

```
}
```

```
});
```

```
,
```

The following example demonstrates default and customized mask placeholder value.

INDEX.JS

```
var datePickerObject = new ej.calendars.DatePicker({
    enableMask: true
});
datePickerObject.appendTo('#mask');
var datePickerPlaceholder = new ej.calendars.DatePicker({
    enableMask: true,
    maskPlaceholder: {day: 'd', month: 'M', year: 'y'}
});
datePickerPlaceholder.appendTo('#placeholder');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DatePicker control</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
  <body>

    <div id="container" style="margin:50px auto 0; width:250px;">
      <br><br>
      <label class="custom-input-label">Default mask placeholder</label>
      <br><br>
      <input id="mask">
      <br><br><br>
      <label class="custom-input-label">Custom mask placeholder</label>
      <br><br>
      <input id="placeholder">
      </div>

  <script>
```

```

var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Globalization in ##Platform_Name## Datepicker control

Globalization is the combination of adapting the component to various languages by means of parsing and formatting the date or number [Internationalization](#) and also by adding cultural specific customizations and translating the text [localization](#)

By default, DatePicker date format, week and month names are specific to English culture. It utilizes the [Essential JavaScript 2 Internationalization](#) package to parse and format the date object based on the culture by using the official [UNICODE CLDR](#) JSON data and it allows to load the culture specific CLDR JSON data by using `loadCldr` method

The DatePicker component supports only the Gregorian type of calendar. All the Essential JS 2 component are specific to English culture ('en-US'). If you want to go with the different culture other than English, follow the below steps.

- Install the `CLDR-Data` package by using the below command (it installs the CLDR JSON data). To know more about CLDR-Data refer the

[CLDR-Data](#) link.

,

```
npm install cldr-data --save
```

,

Once the package installed, you can find the culture specific JSON data under the location `/node_modules/cldr-data`.

- Now import the installed CLDR JSON data into the `app.ts` file. To import JSON data we need to install the JSON plugin loader. Here we have used the SystemJS JSON plugin loader.

,

```
npm install systemjs-plugin-json --save-dev
```

,

- Once installed, configure the `system.config.js` configuration settings as like below to map the `systemjs-plugin-json` loader.

```
`ts
```

```
System.config({
```

```
paths: {
```

```

'npm:': './node_modules/',
'syncfusion:': 'npm:@syncfusion/'
},
map: {
app: 'app',
//Syncfusion packages mapping
"@syncfusion/ej2-base": "syncfusion:ej2-base/dist/ej2-base.umd.min.js",
"@syncfusion/ej2-data": "syncfusion:ej2-data/dist/ej2-data.umd.min.js",
"@syncfusion/ej2-inputs": "syncfusion:ej2-inputs/dist/ej2-inputs.umd.min.js",
"@syncfusion/ej2-popups": "syncfusion:ej2-popups/dist/ej2-popups.umd.min.js",
"@syncfusion/ej2-buttons": "syncfusion:ej2-buttons/dist/ej2-buttons.umd.min.js",
"@syncfusion/ej2-splitbuttons": "syncfusion:ej2-splitbuttons/dist/ej2-splitbuttons.umd.min.js",
"@syncfusion/ej2-lists": "syncfusion:ej2-lists/dist/ej2-lists.umd.min.js",
"@syncfusion/ej2-calendars": "syncfusion:ej2-calendars/dist/ej2-calendars.umd.min.js",
"cldr-data": 'npm:cldr-data',
"plugin-json": "npm:systemjs-plugin-json/json.js"
},
meta: {
'*.json': { loader: 'plugin-json' }
},
packages: {
'app': { main: 'app', defaultExtension: 'js' },
'cldr-data': { main: 'index.js', defaultExtension: 'js' }
}
});
System.import('app');
`

```

- Now use the `loadCldr` method to load the culture specific CLDR JSON data from the installed location to `app.ts` file.
- DatePicker displayed **Sunday** as the first day of week based on default culture ("en-US"). If you want to display the DatePicker with loaded culture's first day of week, you need to import `weekdata.json` file from the `cldr-data/supplemental` as given in the code example.

```

`ts

```

```
//Load the loadCldr from ej2-base
import { loadCldr } from '@syncfusion/ej2-base';
declare var require: any;
loadCldr(
  require('cldr-data/main/de/ca-gregorian.json'),
  require('cldr-data/main/de/numbers.json'),
  require('cldr-data/main/de/timeZoneNames.json'),
  require('cldr-data/supplemental/weekdata.json') // To load the culture based first day of week
);
`
```

The **Localization** library allows you to localize default text content of the DatePicker. The DatePicker component has static text for **today** feature that can be changed to other cultures (Arabic, Deutsch, French, etc.) by defining the [locale](#) value and translation object.

Locale keywords |Text

today | Name of the button to choose Today date.

placeholder | Hint to describe expected value in input element.

- Before changing the culture other than **English**, ensure that locale text for the concerned culture is loaded through **load** method of

[L10n](#) class.

```
`ts
//Load the L10n from ej2-base
import { L10n } from '@syncfusion/ej2-base';
//load the locale object to set the localized placeholder value
L10n.load({
  'de': {
    'datepicker': { placeholder: 'Wählen Sie ein Datum aus',
    today:'heute' }
  }
});
`
```

- Set the culture by using the [locale](#) property. The below code example, initialize the DatePicker component in **German** culture with corresponding localized text.

```

`ts
//Load the L10n from ej2-base
import { L10n } from '@syncfusion/ej2-base';
import { DatePicker } from '@syncfusion/ej2-calendars';
//load the locale object to set the localized placeholder value
L10n.load({
  'de': {
    'datepicker': { placeholder: 'Wählen Sie ein Datum aus',
    today:'heute' }
  }
});
let datePickerObject: DatePicker = new DatePicker({
//sets the locale.
  locale: 'de'
});
datePickerObject.appendTo('#element');
`

```

The following example demonstrates the DatePicker in **German** culture.

INDEX.JS

```

ej.base.enableRipple(true);

var L10n = ej.base.L10n;
L10n.load({
  'de': {
    'datepicker': { placeholder: 'Wählen Sie ein Datum aus',
    today:'heute' }
  }
});
loadCultureFiles('de');
var datePicker = new ej.calendars.DatePicker({
  locale: 'de'
});
datePicker.appendTo('#element');
function loadCultureFiles(name) {
  var files = ['ca-gregorian.json', 'numbers.json',
'timeZoneNames.json'];
  var loader = ej.base.loadCldr;
  var loadCulture = function (prop) {
    var val, ajax;
    ajax = new
ej.base.Ajax('https://ej2.syncfusion.com/javascript/demos/' +
'src/common/cldr-data/main/' + name + '/' + files[prop], 'GET', false);
    ajax.onSuccess = function (value) {

```

```

        val = value;
    };
    ajax.send();
    loader(JSON.parse(val));
};
for (var prop = 0; prop < files.length; prop++) {
    loadCulture(prop);
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 DatePicker control</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <!--style reference from the DatePicker component-->
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <input id="element">
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

Right-To-Left

The DatePicker supports right-to-left functionality for languages like Arabic, Hebrew to displays the text in the right-to-left direction. Use

[enableRtl](#) property to set the RTL direction.

```
`ts
import { DatePicker } from '@syncfusion/ej2-calendars';
//Load the L10n from ej2-base
import { L10n, loadCldr } from '@syncfusion/ej2-base';
declare var require: any;
loadCldr(
  require('cldr-data/supplemental/numberingSystems.json'),
  require('cldr-data/main/he/ca-gregorian.json'),
  require('cldr-data/main/he/numbers.json')
  require('cldr-data/main/he/timeZoneNames.json')
);
//load the locale object to set the localized placeholder value
L10n.load({
  'he': {
    'datepicker': { placeholder: 'הזן תאריך',
    today:'היום' }
  }
});
// creates the datepicker with Hebrew culture.
let datepickerobject: DatePicker = new DatePicker({
  //sets the locale
  locale: 'he',
  value: new Date(),
  //sets the enableRtl
  enableRtl: true
});
datepickerobject.appendTo('#element');
```

The below code example demonstrates the DatePicker component in **Hebrew** culture and also explains how to set the localized text to

the placeholder using **load** method of [L10n](#) class.

INDEX.JS

```

ej.base.enableRipple(true);
loadCultureFiles('he');
var L10n=ej.base.L10n;
L10n.load({
  'he': {
    'datepicker': {
      placeholder: 'הזן תאריך',
      today: 'היום'
    }
  }
});
var datepicker = new ej.calendars.DatePicker({
  locale: 'he',enableRtl:true
});
datepicker.appendTo('#element');

function loadCultureFiles(name) {
  var files = ['ca-gregorian.json', 'numbers.json',
'timeZoneNames.json'];
  if (name === 'ar') {
    files.push('numberingSystems.json');
  }
  var loader = ej.base.loadCldr;
  var loadCulture = function (prop) {
    var val, ajax;
    if (name === 'ar' && prop === files.length - 1) {
      ajax = new
ej.base.Ajax('https://ej2.syncfusion.com/javascript/demos/src/common/cldr-
data/supplemental/' + files[prop], 'GET', false);
    } else {
      ajax = new
ej.base.Ajax('https://ej2.syncfusion.com/javascript/demos/src/common/cldr-
data/main/' + name + '/' + files[prop], 'GET', false);
    }
    ajax.onSuccess = function (value) {
      val = value;
    };
    ajax.send();
    loader(JSON.parse(val));
  };
  for (var prop = 0; prop < files.length; prop++) {
    loadCulture(prop);
  }
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DatePicker control</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <!--style reference from the DatePicker component-->

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <input id="element">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Strict mode in ##Platform_Name## Datepicker control

The [strictMode](#) is an act, that allows the user to enter only the valid date within the specified min/max range in textbox. If the date is invalid, then the component will stay with the previous value. Else, if the date is out of range, then the component will set the date to the min/max date.

The following example demonstrates the DatePicker in `strictMode` with min/max range of 5th to 25th in a month of May. Here, it allows to enter

only the valid date within the specified range. If you are trying to enter the out-of-range value as like 28th of May, then the value will set to the max date of 25th May. Since the value 28th is greater than to max value of 25th. Or else if you are trying to enter the invalid date, then the value will stay with the previous value.

INDEX.JS

```

var datepicker = new ej.calendars.DatePicker({
    placeholder: 'Choose a date',
    // sets the value
    value: new Date('27/10/2017'),
    //sets the min
    min: new Date('5/5/2017'),
    //sets the max
    max: new Date('5/25/2017'),
    // sets the format

```

```

        format: 'dd/MM/yyyy',
    });
    datepicker.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 DatePicker control</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <input id="element" type="text">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

By default, the DatePicker act in strictMode **false** state, that allows to enter the invalid or out-of-range date in textbox.

If the date is out-of-range or invalid, then the model value will be set to **out of range** date value or **null** respectively with highlighted **error** class to indicates the date is out of range or invalid.

The following example demonstrates the **strictMode** as **false**. Here, it allows to enter the valid or invalid value in textbox. If you are entering out-of-range or invalid date value, then the model value will be set to **out of range** date value or **null** respectively with highlighted **error** class to indicates the date is out of range or invalid.

INDEX.JS

```
var datepicker = new ej.calendars.DatePicker({
  placeholder: 'Choose a date',
  // sets the value
  value: new Date('27/10/2017'),
  //sets the min
  min: new Date('5/5/2017'),
  //sets the max
  max: new Date('5/25/2017'),
  // sets the format
  format: 'dd/MM/yyyy',
});
datepicker.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DatePicker control</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <input id="element" type="text">
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

If the value of `min` or `max` properties changed through code behind, then you have to update the `value` property to set within the range.

Customization in ##Platform_Name## Datepicker control

You can customize the entire appearance of the input element and Calendar by using custom [cssClass](#) property.

and also you can use the calendar's [renderDayCell](#) event to customize the appearance of the each day cell.

Below is the list of classes that provides flexible way to customize the DatePicker component.

Class Name	Description
---	---
e-date-wrapper	Applied to DatePicker wrapper
e-datepicker	Applied to the DatePicker element.
e-float-text	Applied to the floating label.
e-date-icon	Applied to the DatePicker icon.
e-popup-wrapper	Applied to DatePicker popup wrapper.
e-calendar	Applied to Calendar element.
e-header	Applied to Calendar header.
e-title	Applied to Calendar title.
e-icon-container	Applied to Calendar previous and next icon container.
e-prev	Applied to Calendar previous icon.
e-next	Applied to Calendar next icon.
e-weekend	Applied to Calendar weekend dates.
e-other-month	Applied to Calendar other month dates.
e-day	Applied to each day cell of the Calendar.
e-selected	Applied to Calendar selected dates.
e-disabled	Applied to Calendar disabled dates.

The following example disables the weekends of every month using `renderDayCell` event. Here we have used the `e-disabled` class to highlight the disabled date.

INDEX.JS

```
var datepicker = new ej.calendars.DatePicker({
    placeholder: 'Choose a date',
    // Bind the renderDayCell event to customize the each day cell.
    renderDayCell: onRenderCell,
    // sets the placeholder
    placeholder: 'Enter date',
    cssClass: 'e-custom-style'
});
```

```
function onRenderCell(args) {
    if (args.date.getDay() == 0 || args.date.getDay() == 6) {
        //sets isDisabled to true to disable the date.
        args.isDisabled = true;
        //To know about the disabled date customization, you can refer in
        "styles.css".
    }
}
datepicker.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Calendar control</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="styles.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <input id="element">
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>
```

Adding mandatory asterisk to placeholder and float label

You can add a mandatory asterisk(*) to placeholder and float label using `.e-input-group.e-control-wrapper.e-float-input .e-float-text::after` class.

INDEX.JS

```

var month = new Date().getMonth();
var fullYear = new Date().getFullYear();

var datepicker = new ej.calendars.DatePicker({
    placeholder: 'Choose a date',
    // Sets the min.
    min: new Date(fullYear, month, 9),
    //Sets the max.
    max: new Date(fullYear, month, 15),
    // Sets the value.
    value: new Date(fullYear, month, 11)
});
datepicker.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DatePicker control</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="asterisk.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <input id="element" type="text">
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```


See Also

- [How to disable the DatePicker control](#)
- [How to set read-only for DatePicker](#)
- [How to customize the DatePicker day header](#)

Date views in ##Platform_Name## Datepicker control

The DatePicker has the following predefined views that provides a flexible way to navigate back and forth to select the date.

| View | Description |

| --- | --- |

| month (default) | Displays the days in a month |

| year | Displays the months in a year |

| decade | Displays the years in a decade |

Start view

You can use the [start](#) property to define the initial rendering view.

The following example demonstrates how to create a DatePicker with **decade** as initial rendering view.

INDEX.JS

```
var datepicker = new ej.calendars.DatePicker({
  placeholder: 'Choose a date',
  //sets the start
  start: 'Decade'
});
datepicker.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DatePicker control</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-calendars/styles/material.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <input id="element" type="text">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Depth view

Define the [depth](#) property to control the view navigation.

Always the depth view has to be smaller than the start view, otherwise the view restriction will be not restricted.

The following example demonstrates how to create a DatePicker that allows users to select a month.

INDEX.JS

```

var datepicker = new ej.calendars.DatePicker({
    placeholder: 'Choose a date',
    //sets the start
    start: 'Decade',
    //sets the depth
    depth: 'Year'
});
datepicker.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 DatePicker control</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <input id="element" type="text">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

To know more about Calendar views refer the Calendar's [Calendar Views](#) section.

Accessibility in ##Platform_Name## Datepicker control

The DatePicker component followed the accessibility guidelines and standards, including [ADA](#), [Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the DatePicker component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2 Support](#) | |

| [Section 508 Support](#) | |

| [Screen Reader Support](#) | |

| [Right-To-Left Support](#) | |

| [Color Contrast](#) | |

| [Mobile Device Support](#) | |

| [Keyboard Navigation Support](#) | |

```
| Accessibility Checker Validation |  |
| Axe-core Accessibility Validation |  |

<style>
.post .post-content img {
display: inline-block;
margin: 0.5em 0;
}
</style>

<div> - All
features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The
component does not meet the requirement.</div>
```

WAI-ARIA attributes

The Web accessibility defines a way to make web content and web applications more accessible to disabled people. It especially helps the dynamic content change and advanced user interface controls developed with Ajax, HTML, JavaScript, and related technologies.

DatePicker provides built-in compliance with the [WAI-ARIA](#) specifications. WAI-ARIA

supports is achieved through the attributes like `aria-expanded`, `aria-disabled`, `aria-activedescendant` applied to the input element.

To know about the accessibility of Calendar refer to the Calendar's [Accessibility](#) section.

It helps to provide information about the widget for assistive technology to the disabled person in screen reader.

- **Aria-expanded:** attributes indicates the state of a collapsible element.
- **Aria-disabled:** attribute indicates the disabled state of this DatePicker component.
- **Aria-activedescendent:** attribute helps in managing the current active child of the DatePicker component.

Keyboard Interaction

You can use the following keys to interact with the DatePicker. The component implements the keyboard navigation support by following the [WAI-ARIA practices](#).

It supports the below list of shortcut keys.

Input Navigation

Before opening the popup, use the below list of keys to control the popup element.

| **Press** | **To do this** |

| --- | --- |

| **Alt + Down Arrow** | **Opens the popup.** |

| **Alt + Up Arrow** | **Closes the popup.** |

| **Esc** | **closes the popup.** |

Calendar Navigation

Use the below list of keys to navigate the Calendar after the popup has opened.

| **Press** | **To do this** |

| --- | --- |

| **Upper Arrow** | **Focus the previous week date.** |

| **Down Arrow** | **Focus the next week date.** |

| **Left Arrow** | **Focus the previous date.** |

| **Right Arrow** | **Focus the next date.** |

| **Home** | **Focus the first date in the month.** |

| **End** | **Focus the last date in the month.** |

| **Page Up** | **Focus the same date in the previous month.** |

| **Page Down** | **Focus the same date in the next month.** |

| **Enter** | **Select the currently focused date.** |

| **Shift + Page Up** | **Focus the same date in the previous year.** |

| **Shift + Page Down** | **Focus the same date in the previous year.** |

| **Control + Upper Arrow** | **Moves into the inner level of view like month-year, year-decade** |

| **Control + Down Arrow** | **Moves out from the depth level view like decade-year, year-month** |

| **Control + Home** | **Focus the starting date in the current year.** |

| **Control + End** | **Focus the ending date in the current year.** |

To focus the DatePicker component use the **alt+t** keys.

INDEX.JS

```
var datepicker = new ej.calendars.DatePicker({
    placeholder: 'Choose a date'
});
document.onkeyup = function (e) {
    if (e.altKey && e.keyCode === 84 /* t */) {
        // press alt+t to focus the component.
        datepicker.element.focus();
    }
};
datepicker.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DatePicker control</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <input id="element" type="text">
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Ensuring accessibility

The DatePicker component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the DatePicker component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the DatePicker component with accessibility tools.

See also

- [Accessibility in Syncfusion components](#)

Style appearance in ##Platform_Name## Datepicker control

The following content provides the exact CSS structure that can be used to modify the control's appearance based on the user preference.

Customizing the appearance of DatePicker wrapper element

Use the following CSS to customize the appearance of wrapper element.

,

/ To specify height and font size /

```
.e-input-group input.e-input, .e-input-group.e-control-wrapper input.e-input {
```

```
height: 40px;
```

```
font-size: 20px;
```

```
}
```

,

Customizing the DatePicker icon element

Use the following CSS to customize the DatePicker icon element

,

/ To specify background color and font size /

```
.e-input-group .e-input-group-icon:last-child, .e-input-group.e-control-wrapper .e-input-group-icon:last-child {
```

```
font-size: 12px;
```

```
background-color: darkgray;
```

```
}
```

,

Customizing the Calendar popup of the DatePicker

Please check the below section, to customize the style and appearance of the Calendar component

[Customizing Calendar's style and appearance](#)

Full screen mode support in mobiles and tablets

The DatePicker component's full-screen mode feature enables users to view the component popup element in full-screen mode on mobile devices with improved visibility and a better user experience. It is important to mention that this feature is exclusively available for mobile and tablet devices in both landscape and portrait orientations. To activate the full screen mode within the DatePicker component, simply set the [fullScreenMode](#) API value to `true`. This action will extend the calendar element to occupy the entire screen on mobile devices.

```
import { DatePicker } from '@syncfusion/ej2-calendars';
```


```
// creates a datepicker with fullScreenMode property
```


```
let datePickerObject: DatePicker = new DatePicker({
```

```
// Enable Full Screen Mode
```

```
fullScreenMode: true,  
});  
datepickerObject.appendTo('#element');  
,
```


Default Sample

5/7/2018 



How To

Disabled the datepicker component in `##Platform_Name##` Datepicker control

To disable the DatePicker, use its [enable](#) property.

The following example demonstrates the DatePicker in a disabled state.

INDEX.JS

```
var datepicker = new ej.calendars.DatePicker({
  placeholder: 'Choose a date',
  // sets the enabled
  enabled: false
});
datepicker.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DatePicker control</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <input id="element" type="text">
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Set the placeholder in `##Platform_Name##` DatePicker control

The following example demonstrates how to set `placeholder` in the DatePicker component.

Using `placeholder` you can display a short hint in the input element.

INDEX.JS

```
var datePicker = new ej.calendars.DatePicker({
    // sets the palceholder property.
    placeholder: 'Enter date'
});
datePicker.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DatePicker control</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <input id="element" type="text">
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Set the `readonly` in `##Platform_Name##` DatePicker control

The following example demonstrates how to set `readonly` in DatePicker component. You can achieve this by using `readonly` property.

INDEX.JS

```
var datePicker = new ej.calendars.DatePicker({
    placeholder: 'Choose a date',
    // sets the readonly.
    readonly: true,
    // sets the value.
    value: new Date()
});
datePicker.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DatePicker control</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <input id="element" type="text">
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Prevent the popup close in ##Platform_Name## DatePicker control

To prevent the DatePicker popup from closing, use the `preventDefault` method from the `PreventableEventArgs`.

The following example demonstrates how to prevent the popup from closing.

INDEX.JS

```
var datePicker = new ej.calendars.DatePicker({
    placeholder: 'Choose a date',
    close: function (args) {
        // prevent the popup close
        args.preventDefault();
    },
});
datePicker.appendTo('#element');
// open the datepicker popup
datePicker.show();
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 DatePicker control</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-calendars/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="container">
        <input id="element" type="text">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
```

```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Customize the datepicker day header in ##Platform_Name## DatePicker control

You can change the format of the day that to be displayed in header using [dayHeaderFormat](#) property.

By default, the format is **Short**.

You can find the possible formats on below.

Name	Description
Short	Sets the short format of day name (like Su) in day header.
Narrow	Sets the single character of day name (like S) in day header.
Abbreviated	Sets the min format of day name (like Sun) in day header.
Wide	Sets the long format of day name (like Sunday) in day header.

INDEX.JS

```
var datePickerObject = new ej.calendars.DatePicker({
    dayHeaderFormat: "Short"
});
datePickerObject.appendTo('#element');
var formatLabel = new ej.dropdowns.DropDownList({
    // set the height of the popup element
    popupHeight: '200px',
    // bind the change event
    change: function(args) {
        datePickerObject.dayHeaderFormat = args.value;
    }
});
formatLabel.appendTo('#select');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
<title>Essential JS 2 DatePicker control</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="styles.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <input id="element">
    </div>
    <div id="format">
        <label class="custom-input-label">Header Format Types</label>
        <div id="wrapper">
            <select id="select" class="form-control">
                <option value="Short" selected="">Short</option>
                <option value="Narrow">Narrow</option>
                <option value="Abbreviated">Abbreviated</option>
                <option value="Wide">Wide</option>
            </select>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Open datepicker popup on input click in ##Platform_Name## Datepicker control

To open the DatePicker popup upon input click by using `show` method in the `focus` event.

The following example demonstrates how to open the DatePicker popup upon focus the input.

INDEX.JS

```

var datepicker = new ej.calendars.DatePicker({
    placeholder: 'Choose a date',
    focus: function() {
        datepicker.show();
    }
});
datepicker.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 DatePicker control</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">

```

```

<meta name="author" content="Syncfusion">
<link href="styles.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <input id="element">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Client side validation in ##Platform_Name## Datepicker control

To achieve the client side validation in a DatePicker component by using [Essential JavaScript 2 FormValidator](#). It provides an option to customize the feedback error messages to the corresponding fields to take action to resolve the issue.

In this below example, the required field validation is implemented by mapping the name attribute value to the rules property. It will validate the DatePicker component and display the validation message when the textbox value is empty during form post back or focus out.

INDEX.JS

```

var datepicker = new ej.calendars.DatePicker({
    placeholder: 'Choose a date',
    // sets the value
    value: new Date()
});
var options = {
    rules: {
        'datevalue': { required: true }
    },
    customPlacement: function (inputElement, errorElement) {
        //to place the error message in custom position.
        inputElement.parentElement.parentElement.appendChild(errorElement);
    }
};

```



```

    }
};
var formObject = new ej.inputs.FormValidator('#form-element', options);
datepicker.appendTo('#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DatePicker control</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <form id="form-element" class="form-vertical">
      <div class="form-group">
        <div class="col-sm-6">
          <input type="text" id="element" name="datevalue"
class="form-control">
        </div>
      </div>
    </form>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

DateRangePicker

Range restriction in ##Platform_Name## Daterangepicker control

Range selection in a date range picker can be made-to-order with desire restrictions based on the application needs.

Restrict the range within a range

You can restrict the minimum and maximum date that can be allowed as start and end date in a range selection with the help of [min](#), [max](#) properties.

- **min** – sets the minimum date that can be selected as startDate.
- **max** – sets the maximum date that can be selected as endDate.

In the following sample, you can select a range from 15th day of this month to 15th day of next month.

INDEX.JS

```
var today = new Date();
var currentYear = today.getFullYear();
var currentMonth = today.getMonth();
var currentDay = today.getDate();
var daterangepicker = new ej.calendars.DateRangePicker({
    //sets the min.
    min: new Date(currentYear, currentMonth, 15),
    //sets the max.
    max: new Date(currentYear, currentMonth+1, 15),
    placeholder:"Select a Range"
});
daterangepicker.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DateRangePicker control</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-calendars/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <input id="element" type="text">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

If the value of **min** or **max** properties changed through code behind, then you have to update the **start date**, **end date** property to set within the range. Or else, if the **start** and **end** date is out of specified date range, a validation error class will be appended to the input element. If **strictMode** is enabled, and both the start, end date is lesser than the min date then start and end date will be updated with **min** date. If both the start and end date is higher than the max date then start and end date will be updated with **max** date. Or else, if **startDate** is less than **min** date, **startDate** will be updated with **min** date or if **endDate** is greater than **max** date, **endDate** will be updated with the **max** date.

Range span

Days span between ranges can be limited in order to avoid excess or less days selection towards the required days in a range. In this, minimum and maximum span allowed within the date range can be customized by [minDays](#) and [maxDays](#) properties.

- **minDays**- Sets the minimum number of days between start and end date.
- **maxDays**- Sets the maximum number of days between start and end date.

In the following sample, the range selection should be greater than 3 days and less than 8 days else it will not set.

INDEX.JS

```

var today = new Date();
var currentYear = today.getFullYear();
var currentMonth = today.getMonth();
var currentDay = today.getDate();
var daterangepicker = new ej.calendars.DateRangePicker({
    //sets the minimum number of days
    minDays: 4,
    //sets the maximum number of days
    maxDays: 7,
    placeholder:"Select a Range"
});
daterangepicker.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DateRangePicker control</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <input id="element" type="text">
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Strict mode

DateRangePicker provides an option to limit the user towards entering the valid date. With strict mode, you can set only the valid date. If any invalid range is specified, the date range value resets to previous value. This restriction can be availed by enabling [strictMode](#) property as true.

INDEX.JS

```

var daterangepicker = new ej.calendars.DateRangePicker({
  placeholder: 'Select a range',
  //enabling strict mode
  strictMode: true
});
daterangepicker.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DateRangePicker control</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <input id="element" type="text">
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Globalization in ##Platform_Name## Daterangepicker control

Globalization is the combination of internalization and localization. You can adapt the component to various languages by parsing and formatting the date or number (Internationalization), and also add culture specific customization and translation to the text (Localization).

By default, DateRangePicker date format, week, and month names are specific to the English culture. It utilizes the [Essential JavaScript 2 Internationalization](#) package to parse and format the date object based on the culture by using the official [UNICODE CLDR](#) JSON data. It allows to load culture specific CLDR JSON data by using `loadCldr` method.

To use a different culture other than `English`, follow the below steps:

- Install the `CLDR-Data` package by using the below command (Installs the CLDR JSON data). To know more about CLDR-Data refer to the [CLDR-Data](#) link.

`ts
npm install cldr-data --save
`

Once the package is installed, you can find the culture specific JSON data under the location `/node_modules/cldr-data`.

- Import the installed CLDR JSON data into the `app.ts` file. To import JSON data, install the JSON plugin loader. Here, The SystemJS JSON plugin loader is used.

`ts
npm install systemjs-plugin-json --save-dev
`

- After installation, configure the `system.config.js` configuration settings as given below to map the `systemjs-plugin-json` loader.

```
`ts
System.config({
  paths: {
    'npm:': './node_modules/',
    'syncfusion:': 'npm:@syncfusion/'
  },
  map: {
    app: 'app',
    //Syncfusion packages mapping
    "@syncfusion/ej2-base": "syncfusion:ej2-base/dist/ej2-base.umd.min.js",
    "@syncfusion/ej2-inputs": "syncfusion:ej2-inputs/dist/ej2-inputs.umd.min.js",
    "@syncfusion/ej2-popups": "syncfusion:ej2-popups/dist/ej2-popups.umd.min.js",
    "@syncfusion/ej2-lists": "syncfusion:ej2-lists/dist/ej2-lists.umd.min.js",
    "@syncfusion/ej2-data": "syncfusion:ej2-data/dist/ej2-data.umd.min.js",
    "@syncfusion/ej2-buttons": "syncfusion:ej2-buttons/dist/ej2-buttons.umd.min.js",
    "@syncfusion/ej2-splitbuttons": "syncfusion:ej2-splitbuttons/dist/ej2-splitbuttons.umd.min.js",
    "@syncfusion/ej2-calendars": "syncfusion:ej2-calendars/dist/ej2-calendars.umd.min.js",
    "cldr-data": 'npm:cldr-data',
    "plugin-json": "npm:systemjs-plugin-json/json.js"
  },
```

```

meta: {
  '*.json': { loader: 'plugin-json' }
},
packages: {
  'app': { main: 'app', defaultExtension: 'js' },
  'cldr-data': { main: 'index.js', defaultExtension: 'js' }
}
});
System.import('app');
`

```

- Use the `loadCldr` method to load the culture specific CLDR JSON data from the installed location to `app.ts` file.
- DateRangePicker displayed `Sunday` as the first day of week based on default culture ("en-US"). If you want to display the DateRangePicker with loaded culture's first day of week, you need to import `weekdata.json` file from the `cldr-data/supplemental` as given in the code example.

```

`ts
declare var require: any;

loadCldr(
  require('cldr-data/main/de/numbers.json'),
  require('cldr-data/main/de/ca-gregorian.json'),
  require('cldr-data/main/de/numbers.json'),
  require('cldr-data/main/de/timeZoneNames.json'),
  require('cldr-data/supplemental/weekdata.json') // To load the culture based first day of week
);
`

```

The `Localization` library allows you to localize default text content of the DateRangePicker. The DateRangePicker component has static text for **today** feature that can be changed to other cultures (Arabic, Deutsch, French, etc.) by defining the [locale](#) value and translation object.

Locale keywords | Text

placeholder | Hint to describe expected value in input element.

startLabel | Label to represent the start date.

endLabel | Label to represent the end date.

applyText | Text present in the apply button.

cancelText | Text present in the cancel button.

selectedDays | Text to represent selected days.

days | Text represents days.

customRange | Text present in the custom range button in presets container.

- Before changing to a culture other than **English**, ensure that locale text for the concerned culture is loaded through **load** method of

[L10n](#) class.

```
`ts
//Load the L10n, loadCldr from ej2-base
import { loadCldr, L10n } from '@syncfusion/ej2-base';
//load the locale object to set the localized placeholder value
L10n.load({
  'de': {
    'daterangepicker': { placeholder: 'Wählen Sie ein Datum aus' }
  }
});
`
```

- Set the culture by using the [locale](#) property. In the following code example, the DateRangePicker is initialized in **German** culture with corresponding localized text.

The following example demonstrates the DateRangePicker in **German** culture.

INDEX.JS

```
ej.base.enableRipple(true);

var L10n = ej.base.L10n;
L10n.load({
  'de': {
    'daterangepicker': {
      placeholder: 'Wählen Sie einen Bereich aus',
      startLabel: 'Wählen Sie Startdatum',
      endLabel: 'Wählen Sie Enddatum',
      applyText: 'Sich bewerben',
      cancelText: 'Stornieren',
      selectedDays: 'Ausgewählte Tage',
      days: 'Tage',
      stomRange: 'benutzerdefinierten Bereich',
      startDate: 'Anfangsdatum',
      endDate: 'Enddatum'
    }
  }
});
loadCultureFiles('de');
var daterangepicker = new ej.calendars.DateRangePicker({
```



```

        locale: 'de'
    });
    daterangepicker.appendTo('#element');
    function loadCultureFiles(name) {
        var files = ['ca-gregorian.json', 'numbers.json',
'timeZoneNames.json'];
        var loader = ej.base.loadCldr;
        var loadCulture = function (prop) {
            var val, ajax;
            ajax = new
ej.base.Ajax('https://ej2.syncfusion.com/javascript/demos/' +
'src/common/cldr-data/main/' + name + '/' + files[prop], 'GET', false);
            ajax.onSuccess = function (value) {
                val = value;
            };
            ajax.send();
            loader(JSON.parse(val));
        };
        for (var prop = 0; prop < files.length; prop++) {
            loadCulture(prop);
        }
    }
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 DateRangePicker control</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <input id="element">
    </div>

```

```

<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Right-To-Left

The DateRangePicker supports RTL (right-to-left) functionality for languages like Arabic and Hebrew to displays the text in the right-to-left direction. Use [enableRtl](#) property to set the RTL direction. The following code example initialize the DateRangePicker component in **Hebrew** culture and also explains how to set the localized text to the placeholder using [load](#) method of

[L10n](#) class.

INDEX.JS

```

ej.base.enableRipple(true);
loadCultureFiles('he');
var L10n = ej.base.L10n;
L10n.load({
    'he': {
        'daterangepicker': {
            placeholder: 'בחר טווח',
            startLabel: 'תווית התחלה',
            endLabel: 'סוף',
            applyText: 'להחיל טקסט',
            cancelText: 'בטל טקסט',
            selectedDays: 'ימים נבחרים',
            days: 'א'ימים',
            customRange: 'טווח מותאם אישית',
            startDate: 'תאריך התחלה',
            endDate: 'תאריך סיום'
        }
    }
});
var daterangepicker = new ej.calendars.DateRangePicker({
    locale: 'he', enableRtl: true
});
daterangepicker.appendTo('#element');

function loadCultureFiles(name) {
    var files = ['ca-gregorian.json', 'numbers.json',
'timeZoneNames.json'];
    if (name === 'he') {
        files.push('numberingSystems.json');
    }
    var loader = ej.base.loadCldr;
    var loadCulture = function (prop) {
        var val, ajax;
        if (name === 'he' && prop === files.length - 1) {
            ajax = new
ej.base.Ajax('https://ej2.syncfusion.com/javascript/demos/src/common/cldr-
data/supplemental/' + files[prop], 'GET', false);

```

```

    } else {
        ajax = new
ej.base.Ajax('https://ej2.syncfusion.com/javascript/demos/src/common/cldr-
data/main/' + name + '/' + files[prop], 'GET', false);
    }
    ajax.onSuccess = function (value) {
        val = value;
    };
    ajax.send();
    loader(JSON.parse(val));
};
for (var prop = 0; prop < files.length; prop++) {
    loadCulture(prop);
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 DateRangePicker control</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <!--style reference from the DateRangePicker component-->
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="container">
        <input id="element">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>

```

```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Customize the date format

Representation of start and end date strings can be customized to required format by using [format](#) property. By default, the format is based on the culture. To know more about the date format standards, refer to the Internationalization Date Format section. In the following sample, the date strings are formatted to `yyyy-MM-dd` and in between the string "to" is set as a [separator](#).

INDEX.JS

```
var daterangepicker = new ej.calendars.DateRangePicker({
  placeholder: 'Select a range',
  format: "yyyy-MM-dd",
  separator: "to"
});
daterangepicker.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DateRangePicker control</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <input id="element" type="text">
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
```

```

}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customization in ##Platform_Name## Daterangepicker control

The DateRangePicker is available for UI customization that can be achieved by using available properties and events in the component.

Day cell format

The DateRangePicker is available for UI customization based on your application requirements. It can be achieved by using [renderDayCell](#) event that provides an option to customize each day cell on rendering.

The following example disables the weekends of every month by using `renderDayCell` event.

INDEX.JS

```

var daterangepicker = new ej.calendars.DateRangePicker({
  placeholder: 'Select a range',
  renderDayCell: onRenderCell,
});
daterangepicker.appendTo('#element');
function onRenderCell(args) {
  if (args.date.getDay() == 0 || args.date.getDay() == 6) {
    //sets isDisabled to true to disable the date.
    args.isDisabled = true;
  }
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DateRangePicker control</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>

```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <input id="element" type="text">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Preset Ranges

DateRangePicker provides an option to set the predefined ranges via [presets](#) property with the corresponding label. This property will accept the values in the order of label, start date (date object), end date (date object), and append these ranges to the presets pop-up for quick selection. In the following sample, you can easily choose the frequently used range options from the list of ranges.

INDEX.JS

```

let today = new Date();
var currentYear = today.getFullYear();
var currentMonth = today.getMonth();
var currentDay = today.getDate();
var daterangepicker = new ej.calendars.DateRangePicker({
    placeholder: 'Select a range',
    presets: [
        { label: 'Today', start: new Date(), end: new Date() },
        { label: 'This Month', start: new Date(new Date().setDate(1)),
end: new Date() },
        { label: 'Last Month', start: new Date(new Date(new
Date().setMonth(new Date().getMonth() - 1)).setDate(1)), end: new Date() },
        { label: 'Last Year', start: new Date(new Date().getFullYear() -
1, 0, 1), end: new Date() },
    ]
});
daterangepicker.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 DateRangePicker control</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <input id="element" type="text">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

First day of week

Start day in a week will differ based on the culture, but you can also customize this based on the application needs. For this, you have to make use of [firstDayOfWeek](#) property. By default, first day of a week in en-US is Sunday. In the following example it is customized to Monday with the help of this property.

INDEX.JS

```

var daterangepicker = new ej.calendars.DateRangePicker({
    placeholder: 'Select a range',
    firstDayOfWeek:1
});
daterangepicker.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
<title>Essential JS 2 DateRangePicker control</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="styles.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <input id="element" type="text">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [How to customize DateRangePicker using cssClass](#)
- [How to disable DateRangePicker control](#)
- [How to customize the DateRangePicker day header](#)

Accessibility in ##Platform_Name## Daterangepicker control

The DateRangePicker component followed the accessibility guidelines and standards, including [ADA](#), [Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the DateRangePicker component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2 Support](#) | |

| [Section 508 Support](#) | |

| [Screen Reader Support](#) | |

| [Right-To-Left Support](#) | |

| [Color Contrast](#) | |

| [Mobile Device Support](#) | |

| [Keyboard Navigation Support](#) | |

| [Accessibility Checker Validation](#) | |

| [Axe-core Accessibility Validation](#) | |

```
<style>
.post .post-content img {
display: inline-block;
margin: 0.5em 0;
}
</style>

<div> - All
features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The
component does not meet the requirement.</div>
```

WAI-ARIA attributes

The web accessibility makes web content and web applications more accessible for people with disabilities. It especially helps in dynamic content change and development of advanced user interface controls with AJAX, HTML, JavaScript, and related technologies. DateRangePicker provides built-in compliance with [WAI-ARIA](#) specifications. WAI-ARIA support is achieved through the attributes like `aria-expanded`, `aria-disabled`, and `aria-activedescendant` applied as an input element.

To know about the accessibility of Calendar, refer to the Calendar's [Accessibility](#) section.

It helps people with disabilities by providing information about the widget for assistive technology in the screen readers. DateRangePicker component contains grid role and grid cell for each day cell.

- **Aria-expanded:** Indicates the currently selected date of the DateRangePicker component.
- **Aria-disabled:** Indicates the disabled state of the DateRangePicker component.

Keyboard Interaction

Use the below keys to interact with the DateRangePicker.

This component implements the keyboard navigation support by following the [WAI-ARIA practices](#).

It supports the following list of shortcut keys:

Input Navigation

Before opening the popup, use the following list of keys to control the popup element.

| **Press** | **To do this** |

| --- | --- |

| **Alt + Down Arrow** | Opens the popup. |

| **Alt + Up Arrow** | Closes the popup. |

| **Esc** | Closes the popup. |

Calendar Navigation

Use the following list of keys to navigate the currently focused Calendar after the popup has opened.

| **Press** | **To do this** |

| --- | --- |

| **Upper Arrow** | Focuses the same day of the previous week. |

| **Down Arrow** | Focuses the same day of the next week. |

| **Left Arrow** | Focuses the day before. |

| **Right Arrow** | Focuses the next day. |

| **Home** | Focuses the first day of the month. |

| **End** | Focuses the last day of the month. |

| **Page Up** | Focuses the same date of the previous month. |

| **Page Down** | Focuses the same date of the next month. |

| **Enter** | Selects the currently focused date. |

| **Shift + Page Up** | Focuses the same date for the previous year. |

| **Shift + Page Down** | Focuses the same date for the next year. |

| **Control + Home** | Focuses the first date of the current year. |

| **Control + End** | Focuses the last date of the current year. |

| **Alt + Right** | Focuses through out the pop-up container in forward direction. |

| **Alt + Left** | Focuses through out the pop-up container in backward direction. |

To focus the DateRangePicker component, use the **alt+t** keys.

INDEX.JS

```

    var daterangepicker = new ej.calendars.DateRangePicker({
        placeholder: 'Select a range'
    });
    daterangepicker.appendTo('#element');
    document.onkeyup = function (e) {
        if (e.altKey && e.keyCode === 84 /* t */) {
            // press alt+t to focus the component.
            daterangepicker.element.focus();
        }
    };
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 DateRangePicker control</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <input id="element" type="text">
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

Ensuring accessibility

The DateRangePicker component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the DateRangePicker component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the DateRangePicker component with accessibility tools.

See also

- [Accessibility in Syncfusion components](#)

Style appearance in ##Platform_Name## Daterangepicker control

The following content provides the exact CSS structure that can be used to modify the control's appearance based on the user preference.

Customizing the appearance of DateRangePicker wrapper element

Use the following CSS to customize the appearance like height and font size of the wrapper element.

`

/ To specify height and font size /

```
.e-input-group input.e-input, .e-input-group.e-control-wrapper input.e-input {  
font-size: 20px;  
height: 40px;  
}
```

`

Customizing the DateRangePicker icon element

Use the following CSS to customize the DateRangePicker icon element

`

/ To specify background color and font size /

```
.e-input-group .e-input-group-icon:last-child, .e-input-group.e-control-wrapper .e-input-group-icon:last-child {  
background-color: darkgray;  
font-size: 14px;  
}
```

`

Customizing the DateRangePicker popup calendar header

Use the following CSS to customize the DateRangePicker popup calendar header

`

/ To specify background and height /

```
.e-daterangepicker.e-popup .e-range-header {
```

```
background: beige;
height: 80px;
}
`
```

Customizing the DateRangePicker popup calendar header title

Use the following CSS to customize the DateRangePicker popup calendar header title

/ To specify color and font size /

```
.e-daterangepicker.e-popup .e-range-header .e-start-label, .e-daterangepicker.e-popup .e-range-header
.e-end-label {
color: brown;
font-size: 30px;
}
`
```

Customizing the DateRangePicker popup calendar content

Use the following CSS to customize the DateRangePicker popup calendar content

/ To specify background color /

```
.e-daterangepicker.e-popup .e-calendar {
background-color: brown;
}
`
```

Customizing the DateRangePicker popup calendar content title

Use the following CSS to customize the DateRangePicker popup calendar content title

/ To specify color and font size /

```
.e-daterangepicker.e-popup .e-calendar .e-header .e-title {
color: beige;
font-size: 20px;
}
`
```

Customizing the DateRangePicker popup calendar previous and next icon

Use the following CSS to customize the DateRangePicker popup calendar previous and next icon

`

/ To specify font size /

```
.e-calendar .e-header .e-prev, .e-calendar .e-header .e-next, .e-bigger.e-small .e-calendar .e-header .e-prev, .e-bigger.e-small .e-calendar .e-header .e-next {
```

```
font-size: 20px;
```

```
}
```

```
,
```

[Customizing the DateRangePicker popup calendar date cell grid on hovering](#)

Use the following CSS to customize the DateRangePicker popup calendar date cell grid on hovering

```
,
```

/ To specify background color and border /

```
.e-calendar .e-content td:hover span.e-day {
```

```
background-color: beige;
```

```
border: 1px solid black;
```

```
}
```

```
,
```

[Customizing the DateRangePicker popup calendar primary button in footer](#)

Use the following CSS to customize the DateRangePicker popup calendar primary button in footer

```
,
```

/ To specify background color and border color /

```
.e-daterangepicker.e-popup .e-footer .e-btn.e-apply.e-flat.e-primary:disabled, .e-daterangepicker.e-popup .e-footer .e-btn.e-apply.e-flat.e-primary:disabled, .e-daterangepicker.e-popup .e-footer .e-css.e-btn.e-apply.e-flat.e-primary:disabled, .e-daterangepicker.e-popup .e-footer .e-css.e-btn.e-apply.e-flat.e-primary:disabled {
```

```
background-color: brown;
```

```
border-color: black;
```

```
}
```

```
,
```

[Customizing the DateRangePicker popup calendar cancel button in footer](#)

Use the following CSS to customize the DateRangePicker popup calendar cancel button in footer

```
,
```

/ To specify background color, color, and border color /

```
.e-daterangepicker.e-popup .e-footer .e-btn.e-flat, .e-daterangepicker.e-popup .e-footer .e-css.e-btn.e-flat {
```

```
background-color: beige;
```

```
border-color: black;
```

```
color: maroon;
}
```

```
,
```

Customizing the footer element in the DateRangePicker popup calendar

Use the following CSS to customize the DateRangePicker popup calendar footer element

```
,
```

/ To specify background color, color, and border color /

```
.e-daterangepicker.e-popup .e-footer {
background-color: beige;
height: 50px;
}
```

```
,
```

Customizing the selected date cell grid in the DateRangePicker popup calendar

Use the following CSS to customize the selected date cell grid in the DateRangePicker popup calendar

```
,
```

/ To specify background and border /

```
.e-calendar .e-content td.e-focused-date.e-today span.e-day {
background: lightgrey;
border: 1px solid black;
}
```

```
,
```

Full screen mode support in mobiles and tablets

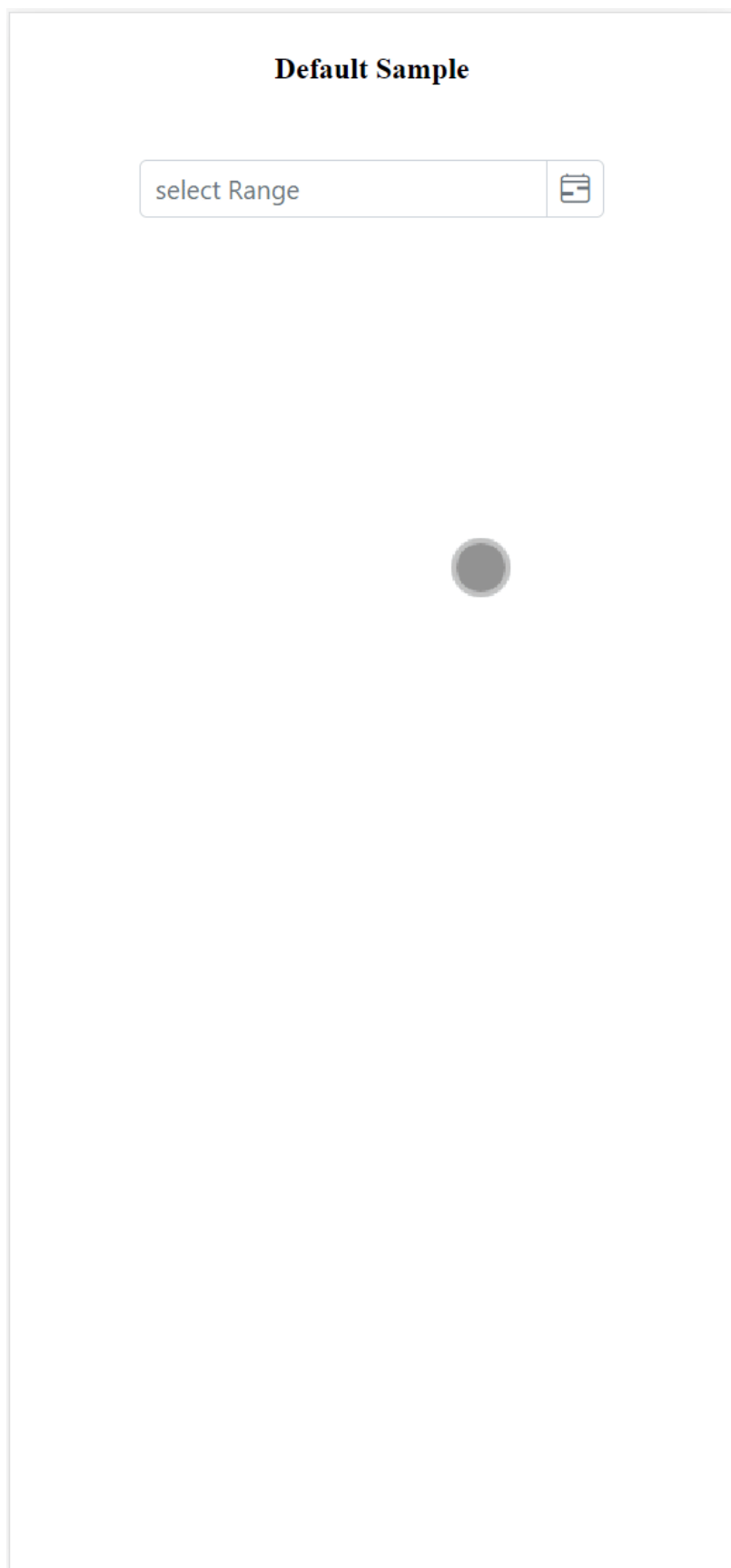
The DateRangePicker component's full-screen mode feature enables users to view the component popup element in full-screen mode on mobile devices with improved visibility and a better user experience. It is important to mention that this feature is exclusively available for mobile and tablet devices in both landscape and portrait orientations. To activate the full screen mode within the DateRangePicker component, simply set the [fullScreenMode](#) API value to `true`. This action will extend the calendar and presets popup element to occupy the entire screen on mobile devices.

```
import { DateRangePicker } from '@syncfusion/ej2-calendars';
// creates a daterangepicker with fullScreenMode property
let daterangeObject: DateRangePicker = new DateRangePicker({
// Enable Full Screen Mode
fullScreenMode: true,
});
daterangeObject.appendTo('#element');
```

DateRangePicker

Style appearance in ##Platform_Name## Daterangepicker control

,



![DateRangePickerPresetsFullScreen](../images/DateRangePickerrPresetsFullScreen.gif)

How To

Disable the `daterangepicker` component in `##Platform_Name## Daterangepicker` control. `DateRangePicker` can be inactivated on a page, by setting `enabled` value as `false` that will disable the component completely from all the user interactions including in form post. The following example demonstrates the disabled component.

INDEX.JS

```
var daterangepicker = new ej.calendars.DateRangePicker({
    // sets the enabled as false
    enabled:false, placeholder:"Select Range"
});
daterangepicker.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DateRangePicker control</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-calendars/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <input id="element" type="text">
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
<script src="index.js" type="text/javascript"></script>
```

```
</body></html>
```

Set the placeholder in `##Platform_Name##` Daterangepicker control

The following example demonstrates how to set [placeholder](#) in the DateRangePicker control.

Using `placeholder` you can display a short hint in the input element.

INDEX.JS

```
var daterangepicker = new ej.calendars.DateRangePicker({
    // sets the placeholder property.
    placeholder: 'Select a range'
});
daterangepicker.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DateRangePicker control</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-calendars/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <input id="element" type="text">
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Customization using cssclass in ##Platform_Name## Daterangepicker control

To customize UI, you can make use of [cssClass](#) that will be added to the DateRangePicker component as the root CSS class. With this CSS class, you can override existing styles of DateRangePicker.

Following is the list of classes that provides flexible way to customize the DateRangePicker component.

Class Name	Description
---	---
e-date-range-wrapper	Applied to DateRangePicker wrapper.
e-range-icon	Applied to DateRangePicker icon.
e-popup	Applied to DateRangePicker popup wrapper.
e-calendar	Applied to both Calendar element.
e-right-calendar	Applied to right Calendar element.
e-left-calendar	Applied to left Calendar element.
e-start-label	Applied to start label in a popup.
e-end-calendar	Applied to end label in a popup.
e-day-span	Applied to day span details label in a popup.
e-footer	Applied to footer elements in a popup.
e-apply	Applied to apply button in footer in a popup.
e-cancel	Applied to cancel button in footer in a popup.
e-header	Applied to Calendar header.
e-title	Applied to Calendar title.
e-icon-container	Applied to Calendar previous and next icon container.
e-prev	Applied to Calendar previous icon.
e-next	Applied to Calendar next icon.
e-weekend	Applied to Calendar weekend dates.
e-other-month	Applied to Calendar other month dates.
e-day	Applied to each day cell of the Calendar.
e-selected	Applied to Calendar selected dates.
e-disabled	Applied to Calendar disabled dates.

INDEX.JS

```
var daterangepicker = new ej.calendars.DateRangePicker({
    // set the css class.
    cssClass:"customCSS", placeholder:"Select Range"
});
daterangepicker.appendTo('#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DateRangePicker control</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <!--style reference from the DateRangePicker component-->
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <input id="element" type="text">
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Customize the daterangepicker day header in ##Platform_Name## Daterangepicker control

You can change the format of the day that to be displayed in header using [dayHeaderFormat](#) property.

By default, the format is **Short**.

You can find the possible formats on below.

Name	Description
----- -----	
Short	Sets the short format of day name (like Su) in day header.
Narrow	Sets the single character of day name (like S) in day header.

| **Abbreviated** | Sets the min format of day name (like Sun) in day header. |

| **Wide** | Sets the long format of day name (like Sunday) in day header. |

INDEX.JS

```
var daterangepickerObj = new ej.calendars.DateRangePicker({
    dayHeaderFormat: "Short",
    cssClass: "format-wide"
});
daterangepickerObj.appendTo('#element');
var formatLabel = new ej.dropdowns.DropDownList({
    // set the height of the popup element
    popupHeight: '200px',
    // bind the change event
    change: function(args) {
        daterangepickerObj.dayHeaderFormat = args.value;
    }
});
formatLabel.appendTo('#select');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DateRngePicker control</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <input id="element">
  </div>
  <div id="format">
    <label class="custom-input-label">Header Format Types</label>
    <div id="wrapper">
```

```

        <select id="select" class="form-control">
            <option value="Short" selected="">Short</option>
            <option value="Narrow">Narrow</option>
            <option value="Abbreviated">Abbreviated</option>
            <option value="Wide">Wide</option>
        </select>
    </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

DateTimePicker

Globalization in ##Platform_Name## Datetimepicker control

Globalization is the combination of internalization and localization. You can adapt the component to various languages by parsing and formatting the date or number [Internationalization](#), and also add culture specific customization and translation to the text [localization](#).

By default, the date format, week, month, time format and meridian names are specific to the [American English](#) culture. It utilizes the

[Essential JavaScript 2 Internationalization](#) package to parse and format the date object based on the culture by using the official [UNICODE CLDR](#) JSON data. It provides the `loadCldr` method to load culture specific CLDR JSON data. To use a different culture other than [English](#), follow the steps below:

- Install the `CLDR-Data` package by using the following command (installs all the CLDR JSON data). To know more about CLDR-Data refer to the [CLDR-Data](#) link.

,

```
npm install cldr-data --save
```

,

Once the package is installed, you can find the culture specific JSON data under the location `/node_modules/cldr-data`.

- Import the installed CLDR JSON data into the `app.ts` file. To import JSON data, install the JSON plugin loader. Here, The SystemJS JSON plugin loader is used.

,

```
npm install systemjs-plugin-json --save-dev
```

,

- After installation, configure the `system.config.js` configuration settings as given below to map the

`systemjs-plugin-json` loader.

```
`ts
System.config({
  paths: {
    'npm:': './node_modules/',
    'syncfusion:': 'npm:@syncfusion/'
  },
  map: {
    app: 'app',
    //Syncfusion packages mapping
    "@syncfusion/ej2-base": "syncfusion:ej2-base/dist/ej2-base.umd.min.js",
    "@syncfusion/ej2-data": "syncfusion:ej2-data/dist/ej2-data.umd.min.js",
    "@syncfusion/ej2-inputs": "syncfusion:ej2-inputs/dist/ej2-inputs.umd.min.js",
    "@syncfusion/ej2-splitbuttons": "syncfusion:ej2-splitbuttons/dist/ej2-splitbuttons.umd.min.js",
    "@syncfusion/ej2-popups": "syncfusion:ej2-popups/dist/ej2-popups.umd.min.js",
    "@syncfusion/ej2-lists": "syncfusion:ej2-lists/dist/ej2-lists.umd.min.js",
    "@syncfusion/ej2-data": "syncfusion:ej2-data/dist/ej2-data.umd.min.js",
    "@syncfusion/ej2-buttons": "syncfusion:ej2-buttons/dist/ej2-buttons.umd.min.js",
    "@syncfusion/ej2-calendars": "syncfusion:ej2-calendars/dist/ej2-calendars.umd.min.js",
    "cldr-data": 'npm:cldr-data',
    "plugin-json": "npm:systemjs-plugin-json/json.js"
  },
  meta: {
    '*.json': { loader: 'plugin-json' }
  },
  packages: {
    'app': { main: 'app', defaultExtension: 'js' },
    'cldr-data': { main: 'index.js', defaultExtension: 'js' }
  }
});
System.import('app');
```


- Use the [loadCldr](#) method to load the culture specific CLDR JSON data from the installed location to `app.ts` file.
- DateTimePicker displayed **Sunday** as the first day of week based on default culture ("en-US"). If you want to display the DateTimePicker with loaded culture's first day of week, you need to import `weekdata.json` file from the `cldr-data/supplemental` as given in the code example.

```
`ts
```

```
import { loadCldr } from '@syncfusion/ej2-base';
declare var require: any;
loadCldr(
  require('cldr-data/main/de/numbers.json'),
  require('cldr-data/main/de/ca-gregorian.json'),
  require('cldr-data/main/de/numbers.json'),
  require('cldr-data/supplemental/numberingSystems.json'),
  require('cldr-data/main/de/timeZoneNames.json'),
  require('cldr-data/supplemental/weekdata.json') // To load the culture based first day of week
);
```

The **Localization** library allows you to localize default text content of the DateTimePicker. The DateTimePicker component has static text for **today** feature that can be changed to other cultures (Arabic, Deutsch, French, etc.) by defining the [locale](#) value and translation object.

Locale keywords | Text

today | Name of the button to choose Today date.

placeholder | Hint to describe expected value in input element.

- Before changing to a culture other than **English**, ensure that locale text for the concerned culture is loaded through `load` method of

[L10n](#) class.

```
`ts
```

```
//Load the L10n, loadCldr from ej2-base
import { loadCldr, L10n } from '@syncfusion/ej2-base';
//load the locale object to set the localized placeholder value
L10n.load({
  'de': {
```

```

'datetimepicker': {
placeholder: 'Wählen Sie ein Datum und eine Uhrzeit aus',
today:'heute'
}
}
});
`

```

- Set the culture by using the [locale](#) property. In the following code example, the DateTimePicker is initialized in **German** culture with corresponding localized text.

The following example demonstrates the DateTimePicker in **German** culture.

INDEX.JS

```

ej.base.enableRipple(true);

var L10n = ej.base.L10n;
L10n.load({
  'de': {
    'datetimepicker': {
      placeholder: 'Wählen Sie ein Datum und eine Uhrzeit aus',
      today: 'heute'
    }
  }
});
loadCultureFiles('de');
var datetimepicker = new ej.calendars.DateTimePicker({
  locale: 'de'
});
datetimepicker.appendTo('#element');
function loadCultureFiles(name) {
  var files = ['ca-gregorian.json', 'numbers.json',
'timeZoneNames.json'];
  var loader = ej.base.loadCldr;
  var loadCulture = function (prop) {
    var val, ajax;
    ajax = new
ej.base.Ajax('https://ej2.syncfusion.com/javascript/demos/' +
'src/common/cldr-data/main/' + name + '/' + files[prop], 'GET', false);
    ajax.onSuccess = function (value) {
      val = value;
    };
    ajax.send();
    loader(JSON.parse(val));
  };
  for (var prop = 0; prop < files.length; prop++) {
    loadCulture(prop);
  }
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DateRangePicker control</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <!--style reference from the DateTimePicker component-->
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-calendars/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <input id="element">
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Right-To-Left

The DateTimePicker supports RTL (right-to-left) functionality for languages like Arabic and Hebrew to displays the text in the right-to-left direction. Use [enableRtl](#) property to set the RTL direction. The following code example initialize the DateTimePicker component in Arabic culture and also explains how to set the localized text to the placeholder using [load](#) method of

[L10n](#) class.

INDEX.JS

```

ej.base.enableRipple(true);
var L10n = ej.base.L10n;
L10n.load({
  'ar': {
    'datetimepicker': {

```

```

        placeholder: 'حدد التاريخ والوقت',
        today: 'اليوم'
    }
    });
    loadCultureFiles('ar');
    var datetimepicker = new ej.calendars.DateTimePicker({
        locale: 'ar'
    });
    datetimepicker.appendTo('#element');
    function loadCultureFiles(name) {
        var files = ['ca-gregorian.json', 'numbers.json',
'timeZoneNames.json'];
        if (name === 'ar') {
            files.push('numberingSystems.json');
        }
        var loader = ej.base.loadCldr;
        var loadCulture = function (prop) {
            var val, ajax;
            if (name === 'ar' && prop === files.length - 1) {
                ajax = new
ej.base.Ajax('https://ej2.syncfusion.com/javascript/demos/src/common/cldr-
data/supplemental/' + files[prop], 'GET', false);
            } else {
                ajax = new
ej.base.Ajax('https://ej2.syncfusion.com/javascript/demos/src/common/cldr-
data/main/' + name + '/' + files[prop], 'GET', false);
            }
            ajax.onSuccess = function (value) {
                val = value;
            };
            ajax.send();
            loader(JSON.parse(val));
        };
        for (var prop = 0; prop < files.length; prop++) {
            loadCulture(prop);
        }
    }
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 DateRangePicker control</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <input id="element">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Strict mode in ##Platform_Name## Datetimepicker control

The [strictMode](#) is an act, that allows the user to enter only the valid date and time within the specified min/max range in textbox. If the input entered is invalid, then the component will stay with the previous value. Else, if the date and time is

out of range, then the component will set the date to the min/max value.

The following example demonstrates the DateTimePicker in `strictMode` with min/max range of 5/5/2019 2:00 AM to 5/25/2019 2:00 AM. Here, it allows to enter only the valid date and time within the specified range. If you are trying to enter the out-of-range value as

like 5/28/2019, then the value will set to the `max` value as 5/25/2019 2:00 AM. Since the value 28 is greater than to `max` value

of 25. Or else if you are trying to enter the invalid date, then the value will stay with the previous value.

INDEX.JS

```

var datetimepickerObject = new ej.calendars.DateTimePicker({
    // sets the value
    value: new Date('5/25/2017 4:00 PM'),
    //sets the min
    min: new Date('5/5/2017 2:00 PM'),
    //sets the max
    max: new Date('5/25/2017 3:00 PM'),
    // sets the placeholder
    placeholder: 'Select a date and time'
});
datetimepickerObject.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DateTimePicker control</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <!--style reference from the DateTimePicker component-->
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <input id="element" type="text">
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

By default, the DateTimePicker act in strictMode **false** state, that allows to enter the invalid or out-of-range datetime in textbox.

If the datetime is out-of-range or invalid, then the model value will be set to **out of range** datetime value or **null** respectively with highlighted **error** class to indicates the datetime is out of range or invalid.

The following example demonstrates the strictMode as **false**. Here, it allows to enter the valid or invalid value in textbox. If you are entering the out-of-range or invalid datetime value, then the model value will be set to **out of range** datetime value or **null** respectively with highlighted **error** class to indicates the datetime is out of range or invalid.

INDEX.JS

```
var datetimepickerObject = new ej.calendars.DateTimePicker({
```

```
// sets the value
value: new Date('5/25/2017 4:00 PM'),
//sets the min
min: new Date('5/5/2017 2:00 PM'),
//sets the max
max: new Date('5/25/2017 3:00 PM'),
// sets the placeholder
placeholder: 'Select a date and time'
});
datetimepickerObject.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DateTimePicker control</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <!--style reference from the DateTimePicker component-->
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-calendars/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <input id="element" type="text">
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

If the value of `min` or `max` properties changed through code behind, then you have to update the `value` property to set within the range.

Date time range in ##Platform_Name## Datetimepicker control

DateTimePicker provides an option to select a date and time value within a specified range by using the [min](#)

and [max](#) properties. Always the min value has to be lesser than the max value.

When the min and max properties are configured and the selected datetime value is out-of-range or invalid, then the model value will be set to `out of range` datetime value or `null` respectively with highlighted `error` class to indicates the datetime is out of range or invalid.

The value property depends on the min/max with respect to [strictMode](#) property.

The below example allows selecting a date within the range from 7th to 27th day in a month.

INDEX.JS

```
ej.base.enableRipple(true);
var today = new Date();
var currentYear = today.getFullYear();
var currentMonth = today.getMonth();
var currentDay = today.getDate();
var datetimepicker = new ej.calendars.DateTimePicker({
  min: new Date(currentYear, currentMonth, 7, 10),
  max: new Date(currentYear, currentMonth, 27, 22, 30),
  value: new Date(currentYear, currentMonth, 14, 10, 30),
  placeholder: 'Choose a datetime'
});
datetimepicker.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DateTimePicker control</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <!--style reference from the DateTimePicker component-->
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-calendars/styles/material.css" rel="stylesheet">
```



```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <input id="element" type="text">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

If the value of `min` or `max` properties changed through code behind, then you have to update the `value` property to set within the range.

Date time format in ##Platform_Name## Datetimepicker control

DateTime format is a way of representing the date and time value in different string format in the textbox.

By default, the DateTimePicker's format is based on the culture. You can also set the own custom format by using the [format](#) property.

Once the format property has been defined it will be common to all the cultures.

To know more about the date format standards, refer to the [Internationalization Date Time Format](#) section.

The following example demonstrates the DateTimePicker with the custom format (yyyy-MM-dd hh:mm).

INDEX.JS

```

var datetimepicker = new ej.calendars.DateTimePicker({
    placeholder: 'Choose a date',
    value: new Date(),
    // sets the format.
    format: 'yyyy-MM-dd hh:mm'
});
datetimepicker.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 DateTimePicker control</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">

```

```

<link href="styles.css" rel="stylesheet">
<!--style reference from the DateTimePicker component-->
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <input id="element" type="text">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Date time masking in ##Platform_Name## Datetimepicker control

DateTimePicker has `enableMask` property that provides the option to enable the built-in date masking support. Also, you must inject the `MaskedDateTime` module to enable the masking support.

INDEX.JS

```

var datetimepickerObject = new ej.calendars.DateTimePicker({
    enableMask: true
});
datetimepickerObject.appendTo('#mask');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
<title>Essential JS 2 DateTimePicker control</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="styles.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <input id="mask">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The mask pattern is defined based on the provided date format to the component. If the format is not specified, the mask pattern is formed based on the default format of the current culture.

| **Keys | Actions |**

| --- | --- |

| Up / Down arrows | To increment and decrement the selected portion of the date and time. |

| Left / Right arrows and Tab | To navigate the selection from one portion to next portion |

The following example demonstrates default and custom format of DateTimePicker component with mask.

INDEX.JS

```

var datetimepickerObject = new ej.calendars.DateTimePicker({
    enableMask: true
});
datetimepickerObject.appendTo('#mask');
var datetimepickerFormat = new ej.calendars.DateTimePicker({
    enableMask: true,
    format: "M/d/yyyy hh:mm a",
});

```

```
datetimepickerFormat.appendTo('#format');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DateTimePicker control</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
  <body>

    <div id="container" style="margin:50px auto 0; width:250px;">
      <br><br>
      <label class="custom-input-label">Mask support with default
format</label>
      <br><br>
      <input id="mask">
      <br><br><br>
      <label class="custom-input-label">Mask support with custom
format</label>
      <br><br>
      <input id="format">
    </div>

    <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
    </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Configure Mask Placeholder

You can change mask placeholder value through property `maskPlaceholder`. By default , it takes the full name of date and time co-ordinates such as `day`, `month`, `year`, `hour` etc.

While changing to a culture other than `English`, ensure that locale text for the concerned culture is loaded through load method of `L10n` class for mask placeholder values like below.

```
`ts
//Load the L10n from ej2-base
import { L10n } from '@syncfusion/ej2-base';
//load the locale object to set the localized mask placeholder value
L10n.load({
  'de': {
    'datetimepicker': { day: 'Tag' , month: 'Monat', year: 'Jahr', hour: 'Stunde' ,minute: 'Minute',
    second:'Sekunden' }
  }
});
`
```

The following example demonstrates default and customized mask placeholder value.

INDEX.JS

```
var datetimepickerObject = new ej.calendars.DateTimePicker({
  enableMask: true
});
datetimepickerObject.appendTo('#mask');
var datetimeplaceholder = new ej.calendars.DateTimePicker({
  enableMask: true,
  maskPlaceholder: {day: 'd', month: 'M', year: 'y', hour: 'h', minute:
  'm', second: 's'}
});
datetimeplaceholder.appendTo('#placeholder');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DateTimePicker control</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin:50px auto 0; width:250px;">
        <br><br>
        <label class="custom-input-label">Default mask placeholder</label>
        <br><br>
        <input id="mask">
        <br><br><br>
        <label class="custom-input-label">Custom mask placeholder</label>
        <br><br>
        <input id="placeholder">
    </div>

<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customization in ##Platform_Name## Datetimepicker control

The DateTimePicker is available for UI customization that can be achieved by using available properties and events in the component.

Day and Time Cell format

The DateTimePicker is available for UI customization based on your application requirements. It can be achieved by using [renderDayCell](#) event that provides an option to customize each day cell on rendering.

The following example disables the weekends of every month by using `renderDayCell` event.

INDEX.JS

```

var datetimepicker = new ej.calendars.DateTimePicker({
    placeholder: 'Select a date and time'
});
datetimepicker.appendTo('#element');
document.onkeyup = function (e) {
    if (e.altKey && e.keyCode === 84 /* t */) {
        // press alt+t to focus the component.
        datetimepicker.element.focus();
    }
}

```

```

    }
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DateTimePicker control</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <!--style reference from the DateTimePicker component-->
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-calendars/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <input id="element" type="text">
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Adding mandatory asterisk to placeholder and float label

You can add a mandatory asterisk(*) to placeholder and float label using `.e-input-group.e-control-wrapper.e-float-input.e-float-text::after` class.

INDEX.JS

```

var datetimepicker = new ej.calendars.DateTimePicker({
  placeholder: 'Select DateTime'
});

```

```
datetimepicker.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DateTimePicker control</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="asterisk.css" rel="stylesheet">
  <!--style reference from the DateTimePicker component-->
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <input id="element" type="text">
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

See Also

- [How to disable the DateTimePicker control](#)
- [How to customize the DateTimePicker day header](#)

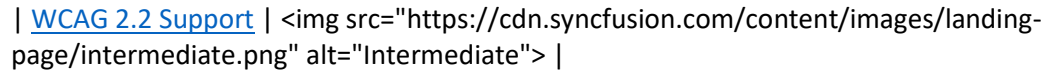
Accessibility in ##Platform_Name## Datetimepicker control

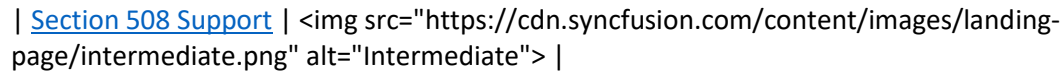
The DateTimePicker component followed the accessibility guidelines and standards, including [ADA](#), [Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

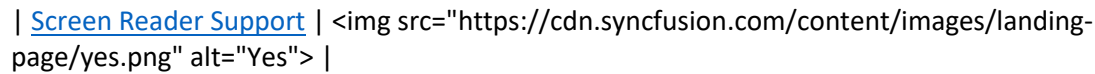
The accessibility compliance for the DateTimePicker component is outlined below.

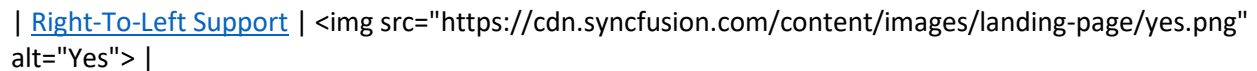
| Accessibility Criteria | Compatibility |

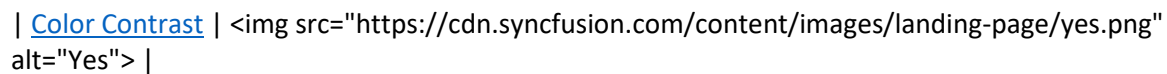
| -- | -- |

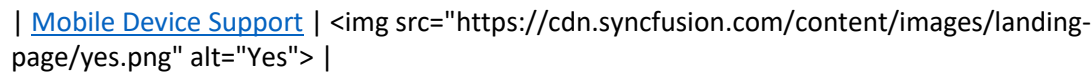
| [WCAG 2.2 Support](#) |  |

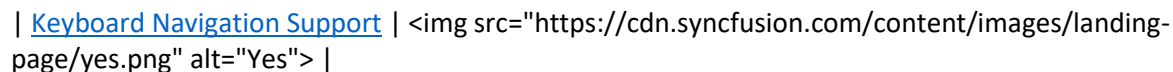
| [Section 508 Support](#) |  |

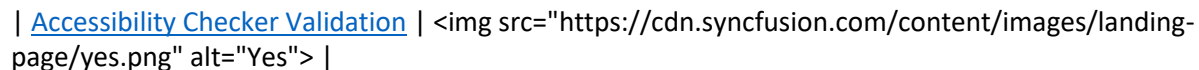
| [Screen Reader Support](#) |  |

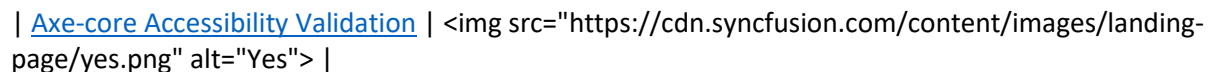
| [Right-To-Left Support](#) |  |

| [Color Contrast](#) |  |

| [Mobile Device Support](#) |  |

| [Keyboard Navigation Support](#) |  |

| [Accessibility Checker Validation](#) |  |

| [Axe-core Accessibility Validation](#) |  |

<style>

```
.post .post-content img {
display: inline-block;
margin: 0.5em 0;
}
```

</style>

<div> - All features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>

WAI-ARIA attributes

The Web accessibility defines a way to make web content and web applications more accessible to disabled people. It especially helps the dynamic content change and advanced user interface controls developed with Ajax, HTML, JavaScript, and related technologies.

DateTimePicker provides built-in compliance with the [WAI-ARIA](#) specifications. WAI-ARIA supports is achieved through the attributes like `aria-expanded`, `aria-disabled`, `aria-activedescendant` applied to the input element.

To know about the accessibility of Calendar refer to the Calendar's [Accessibility](#) section.

It helps to provide information about the widget for assistive technology to the disabled person in screen reader.

- **Aria-expanded:** attributes indicates the state of a collapsible element.
- **Aria-disabled:** attribute indicates the disabled state of this DateTimePicker component.
- **Aria-activedescendent:** attribute helps in managing the current active child of the DateTimePicker component.

Keyboard Interaction

You can use the following keys to interact with the DateTimePicker. The component implements the keyboard navigation support by following the [WAI-ARIA practices](#).

DateTimePicker supports the below list of shortcut keys.

Input Navigation

Before opening the popup, use the below list of keys to `DateTimePicker` control the popup element.

| **Press** | **To do this** |

| --- | --- |

| **Alt + Down Arrow** | **Open the select popup** |

| **Alt + Down Arrow + Alt + Down Arrow** | **Toggle between two popups** |

Calendar Navigation

Use the below list of keys to interact with the Calendar after the DatePicker popup has opened.

| **Press** | **To do this** |

| --- | --- |

| **Upper Arrow** | **Focus the previous week date.** |

| **Down Arrow** | **Focus the next week date.** |

| **Left Arrow** | **Focus the previous date.** |

| **Right Arrow** | **Focus the next date.** |

| **Home** | **Focus the first date in the month.** |

| **End** | **Focus the last date in the month.** |

| **Page Up** | **Focus the same date in the previous month.** |

| **Page Down** | **Focus the same date in the next month.** |

| **Enter** | **Select the currently focused date.** |

| **Shift + Page Up** | **Focus the same date in the previous year.** |

| Shift + Page Down | Focus the same date in the previous year. |

| Control + Upper Arrow | Moves into the inner level of view like month-year, year-decade |

| Control + Down Arrow | Moves out from the depth level view like decade-year, year-month |

| Control + Home | Focus the starting date in the current year. |

| Control + End | Focus the ending date in the current year. |

Use the below list of shortcut keys to interact with the TimePicker after the TimePicker Popup has opened.

| Press | To do this |

| --- | --- |

| Upper Arrow | Navigate and select the previous item. |

| Down Arrow | Navigate and select the next item. |

| Left Arrow | Move the cursor towards arrow key pressed direction. |

| Right Arrow | Move the cursor towards arrow key pressed direction. |

| Home | Navigate and select the first item. |

| End | Navigate and select the last item. |

| Enter | Select the currently focused item and close the popup. |

| Alt + Upper Arrow | Close the popup. |

| Alt + Down Arrow | Open the popup. |

| Esc | Close the popup. |

To focus the DateTimePicker component use the `alt+t` keys.

INDEX.JS

```
var datetimepicker = new ej.calendars.DateTimePicker({
  placeholder: 'Select a date and time'
});
datetimepicker.appendTo('#element');
document.onkeyup = function (e) {
  if (e.altKey && e.keyCode === 84 /* t */) {
    // press alt+t to focus the component.
    datetimepicker.element.focus();
  }
};
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DateTimePicker control</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
```

```

<link href="styles.css" rel="stylesheet">
<!--style reference from the DateTimePicker component-->
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <input id="element" type="text">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Ensuring accessibility

The DateTimePicker component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the DateTimePicker component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the DateTimePicker component with accessibility tools.

See also

- [Accessibility in Syncfusion components](#)

Style appearance in ##Platform_Name## Datetimepicker control

The following content provides the exact CSS structure that can be used to modify the control's appearance based on the user preference.

Customizing the appearance of DateTimePicker wrapper element

Use the following CSS to customize the appearance of wrapper element.

`

/ To specify height and font size /

```
.e-input-group input.e-input, .e-input-group.e-control-wrapper input.e-input {  
font-size: 20px;  
height: 40px;  
}  
`
```

Customizing the DateTimePicker icons element

Use the following CSS to customize the DateTimePicker icons element

`

/ To specify background color and font size /

```
.e-datetime-wrapper .e-input-group-icon.e-date-icon, .e-datetime-wrapper .e-input-group-icon.e-time-  
icon {  
font-size: 16px;  
background-color: blanchedalmond;  
}  
`
```

Customizing the time picker popup in the DateTimePicker

Use the following CSS to customize the time picker popup in the DateTimePicker

`

/ To specify height /

```
.e-datetimepicker.e-popup {  
height: 100px;  
}  
`
```

Customizing the Calendar popup of the DateTimePicker

Please check the below section, to customize the style and appearance of the Calendar component in the DateTimePicker



[Customizing Calendar's style and appearance](#)

Full screen mode support in mobiles and tablets

The DateTimePicker component's full-screen mode feature enables users to view the component popup element in full-screen mode on mobile devices with improved visibility and a better user experience. It is important to mention that this feature is exclusively available for mobile and tablet devices in both landscape and portrait orientations. To activate the full screen mode within the DateTimePicker component, simply set the [fullScreenMode](#) API value to `true`. This action will extend the calendar and time popup element to occupy the entire screen on mobile devices.

```
import { DateTimePicker } from '@syncfusion/ej2-calendars';  
// creates a datetimepicker with fullScreenMode property  
let datetimepickerObject: DateTimePicker = new DateTimePicker({  
  // Enable Full Screen Mode  
  fullScreenMode: true,  
});  
datetimepickerObject.appendTo('#element');  
`
```

Default Sample

12/15/2017 2:00 PM		
--------------------	---	---

How To

Disable the datetimepicker component in ##Platform_Name## Datetimepicker control

To disable the DateTimePicker, use its [enable](#) property to `false`.

INDEX.JS

```
ej.base.enableRipple(true);
var today = new Date();
var currentYear = today.getFullYear();
var currentMonth = today.getMonth();
var currentDay = today.getDate();
var datetimepicker = new ej.calendars.DateTimePicker({
  min: new Date(currentYear, currentMonth, 7, 10),
  max: new Date(currentYear, currentMonth, 27, 22, 30),
  value: new Date(currentYear, currentMonth, 14, 10, 30),
  placeholder: 'Choose a datetime'
});
datetimepicker.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DateTimePicker control</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <!--style reference from the DateTimePicker component-->
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-calendars/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <input id="element" type="text">
  </div>
</script>
var ele = document.getElementById('container');
if(ele) {
```



```

    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Set the placeholder in `##Platform_Name##` Datetimepicker control

The following example demonstrates how to set [placeholder](#) in the DateTimePicker component.

Using `placeholder` you can display a short hint in the input element.

INDEX.JS

```

ej.base.enableRipple(true);
var today = new Date();
var currentYear = today.getFullYear();
var currentMonth = today.getMonth();
var currentDay = today.getDate();
var datetimepicker = new ej.calendars.DateTimePicker({
    min: new Date(currentYear, currentMonth, 7, 10),
    max: new Date(currentYear, currentMonth, 27, 22, 30),
    value: new Date(currentYear, currentMonth, 14, 10, 30),
    placeholder: 'Choose a datetime'
});
datetimepicker.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 DateTimePicker control</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <!--style reference from the DateTimePicker component-->
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-calendars/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

```

```

    <div id="container">
      <input id="element" type="text">
    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customize the datetimepicker day header in ##Platform_Name## Datetimepicker control

You can change the format of the day that to be displayed in header using [dayHeaderFormat](#) property. By default, the format is **Short**.

You can find the possible formats on below.

Name	Description
----- -----	
Short	Sets the short format of day name (like Su) in day header.
Narrow	Sets the single character of day name (like S) in day header.
Abbreviated	Sets the min format of day name (like Sun) in day header.
Wide	Sets the long format of day name (like Sunday) in day header.

INDEX.JS

```

var datetimepickerObj = new ej.calendars.DateTimePicker({
  dayHeaderFormat: "Short"
});
datetimepickerObj.appendTo('#element');
var formatLabel = new ej.dropdowns.DropDownList({
  // set the height of the popup element
  popupHeight: '200px',
  // bind the change event
  change: function(args) {
    datetimepickerObj.dayHeaderFormat = args.value;
  }
});
formatLabel.appendTo('#select');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DateTimePicker control</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <input id="element">
    </div>
    <div id="format">
        <label class="custom-input-label">Header Format Types</label>
        <div id="wrapper">
            <select id="select" class="form-control">
                <option value="Short" selected="">Short</option>
                <option value="Narrow">Narrow</option>
                <option value="Abbreviated">Abbreviated</option>
                <option value="Wide">Wide</option>
            </select>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Diagram

Getting started in ##Platform_Name## Diagram control

This section explains the steps required to create a simple diagram and demonstrates the basic usage of the diagram control.

<!-- markdownlint-disable MD033 -->

Dependencies

The following list of dependencies are required to use the **Diagram** component in your application.

```
|-- @syncfusion/ej2-diagrams
|-- @syncfusion/ej2-base
|-- @syncfusion/ej2-data
|-- @syncfusion/ej2-navigations
|-- @syncfusion/ej2-inputs
|-- @syncfusion/ej2-popups
|-- @syncfusion/ej2-buttons
|-- @syncfusion/ej2-lists
|-- @syncfusion/ej2-splitbuttons
\
```

Installation and Configuration

- To get started with the diagram component, clone the [Essential JS 2 quickStart](#) project and install necessary packages by using the following commands.

```
git clone https://github.com/syncfusion/ej2-quickstart.git quickstart
cd quickstart
npm install
\
```

- Diagram packages should be mapped in the `system.config.js` configuration file.

```
System.config({
  paths: {
    'syncfusion:': './node_modules/@syncfusion/',
  },
  map: {
    app: 'app',
    //Syncfusion packages mapping
    "@syncfusion/ej2-base": "syncfusion:ej2-base/dist/ej2-base.umd.min.js",
    "@syncfusion/ej2-data": "syncfusion:ej2-data/dist/ej2-data.umd.min.js",
    "@syncfusion/ej2-navigations": "syncfusion:ej2-navigations/dist/ej2-navigations.umd.min.js",
    "@syncfusion/ej2-inputs": "syncfusion:ej2-inputs/dist/ej2-inputs.umd.min.js",
    "@syncfusion/ej2-popups": "syncfusion:ej2-popups/dist/ej2-popups.umd.min.js",
    "@syncfusion/ej2-buttons": "syncfusion:ej2-buttons/dist/ej2-buttons.umd.min.js",
  }
});
```

```
"@syncfusion/ej2-lists": "syncfusion:ej2-lists/dist/ej2-lists.umd.min.js",
"@syncfusion/ej2-splitbuttons": "syncfusion:ej2-splitbuttons/dist/ej2-splitbuttons.umd.min.js",
"@syncfusion/ej2-diagrams": "syncfusion:ej2-diagrams/dist/ej2-diagrams.umd.min.js",
},
packages: {
  'app': { main: 'app', defaultExtension: 'js' }
}
});
`
```

The [project](#) is preconfigured with common settings (`src/styles/styles.css`, `system.config.js`) to start with all Essential JS 2 components.

Add diagram to the project

Add the HTML div element for the diagram into your `index.html`. [`src/index.html`]

```
`html
<!DOCTYPE html>
<html lang="en">
<head>
<title>EJ2 Diagram</title>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<meta name="description" content="Typescript UI Controls" />
<meta name="author" content="Syncfusion" />
<link href="index.css" rel="stylesheet" />
<script src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></script>
<script src="systemjs.config.js"></script>
</head>
<body>
<!--container which is going to render the Diagram-->
<div id='container'>
</div>
</body>
</html>
`
```

Now, import the diagram component into your `app.ts` to instantiate a diagram and append the diagram instance to the `#container`. [src/app/app.ts]

The following example shows a basic diagram.

INDEX.JS

```
var diagram = new ej.diagrams.Diagram({ width: '100%', height: '600px' });
diagram.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Now, the `npm run start` command is used to run the application in the browser.

npm run start

,

Module Injection

The diagram component is divided into individual feature-wise modules. In order to use a particular feature, inject the required module. The following list describes the module names and their description.

- **BpmnDiagrams**: Inject this provider to add built-in BPMN shapes to diagrams.
- **ConnectorBridging**: Inject this provider to add bridges to connectors.
- **ConnectorEditing**: Inject this provider to edit the segments for connector.
- **ComplexHierarchicalTree**: Inject this provider to complex hierarchical tree like structure.
- **DataBinding**: Inject this provider to populate nodes from given data source.
- **DiagramContextMenu**: Inject this provider to manipulate context menu.
- **HierarchicalTree**: Inject this provider to use hierarchical tree like structure.
- **LayoutAnimation**: Inject this provider animation to layouts.
- **MindMap**: Inject this provider to use mind map.
- **PrintAndExport**: Inject this provider to print or export the objects.
- **RadialTree**: Inject this provider to use radial tree like structure.
- **Snapping**: Inject this provider to snap the objects.
- **SymmetricLayout**: Inject this provider to render layout in symmetrical method.
- **UndoRedo**: Inject this provider to revert and restore the changes.

These modules should be imported and injected into the Diagram component using **Diagram.Inject** method as follows.

```
import { Diagram, HierarchicalTree, MindMap, RadialTree, ComplexHierarchicalTree, DataBinding, Snapping, PrintAndExport, BpmnDiagrams, SymmetricLayout, ConnectorBridging, UndoRedo, LayoutAnimation, DiagramContextMenu, ConnectorEditing } from '@syncfusion/ej2-diagrams';
```

```
Diagram.Inject(BpmnDiagrams, ConnectorBridging, ConnectorEditing, ComplexHierarchicalTree, DataBinding, DiagramContextMenu, HierarchicalTree, LayoutAnimation, MindMap, PrintAndExport, RadialTree, Snapping, SymmetricLayout, UndoRedo);
```

,

Flow Diagram

Create and add node

Create and add a **node** (JSON data) with specific position, size, label, and shape.

INDEX.JS

```
var nodes = [
  {
    id: 'Start', width: 140, height: 50, offsetX: 300, offsetY: 50,
    annotations: [{
      id: 'label1',
      content: 'Start'
    }],
    shape: { type: 'Flow', shape: 'Terminator' }
  }
]
```

```

];
var diagram = new ej.diagrams.Diagram({
  width: '100%', height: '600px', nodes: nodes
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Note: The **annotations** property is an array, which indicates that more than one label can be added to a node.

Connect Nodes

Add two node to the diagram as shown in the previous example. Connect these nodes by adding a connector using the **connector** property and refer the source and target end by using the **sourceNode** and **targetNode** properties.

INDEX.JS

```

var nodes = [
  {
    id: 'Start', width: 140, height: 50, offsetX: 300, offsetY: 50,
    annotations: [{
      id: 'label1',
      content: 'Start'
    }],
    shape: { type: 'Flow', shape: 'Terminator' }
  },
  {
    id: 'Init', width: 140, height: 50, offsetX: 300, offsetY: 140,
    shape: { type: 'Flow', shape: 'process' },
    annotations: [{ content: 'var i = 0;' }]
  }
];
var connector = {
  id: "connector1",
  sourceID: "Start",
  targetID: "Init",
  type: 'Orthogonal'
};
var diagram = new ej.diagrams.Diagram({
  width: '100%', height: '350px', nodes: nodes, connectors: [connector]
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```

```

    <div id="container">
      <div id="element"></div>
    </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Default values for all **nodes** and **connectors** can be set using the **getNodeDefaults** and **getConnectorDefaults** properties, respectively. For example, if all nodes have the same width and height, such properties can be moved into **getNodeDefaults**.

[Complete flow diagram](#)

Similarly, the required nodes and connectors can be added to form a complete flow diagram.

INDEX.JS

```

var nodes = [
  { id: 'Start', offsetY: 50, annotations: [{ content: 'Start' }], shape:
  { type: 'Flow', shape: 'Terminator' } },
  { id: 'Init', offsetY: 140, annotations: [{ content: 'var i = 0;' }],
  shape: { type: 'Flow', shape: 'Process' } },
  { id: 'Condition', offsetY: 230, annotations: [{ content: 'i < 10?' }],
  shape: { type: 'Flow', shape: 'Decision' } },
  { id: 'Print', offsetY: 320, annotations: [{ content:
  'print(\'Hello!!\');' }], shape: { type: 'Flow', shape: 'PreDefinedProcess'
  } },
  { id: 'Increment', offsetY: 410, annotations: [{ content: 'i++;' }],
  shape: { type: 'Flow', shape: 'Process' } },
  { id: 'End', offsetY: 500, annotations: [{ content: 'End' }], shape: {
  type: 'Flow', shape: 'Terminator' } },
];
var connector = [
  { id: 'connector1', sourceID: 'Start', targetID: 'Init' },
  { id: 'connector2', sourceID: 'Init', targetID: 'Condition' },
  { id: 'connector3', sourceID: 'Condition', targetID: 'Print',
  annotations: [{ content: 'Yes' }] },
  {
    id: 'connector4', sourceID: 'Condition', targetID: 'End',
    annotations: [{ content: 'No' }],
    type: 'Orthogonal',
    segments: [
      { type: 'Orthogonal', length: 30, direction: "Right" },
      { type: 'Orthogonal', length: 300, direction: "Bottom" }
    ]
  },
  { id: 'connector5', sourceID: 'Print', targetID: 'Increment' },
  {
    id: 'connector6', sourceID: 'Increment', targetID: 'Condition',
    type: 'Orthogonal',
    segments: [

```

```

        { type: 'Orthogonal', length: 30, direction: "Left" },
        { type: 'Orthogonal', length: 200, direction: "Top" }
    ]
}
];
var diagram = new ej.diagrams.Diagram({
    width: '100%', height: '600px', nodes: nodes, connectors: connector
}, '#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Diagram</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>
```

Automatic organization chart

In the 'Flow Diagram' section, how to create a diagram manually was discussed. This section explains how to create and position the diagram automatically.

Business object (Employee information)

Define Employee Information as JSON data. The following code example shows an employee array whose, `Name` is used as a unique identifier and `ReportingPerson` is used to identify the person to whom an employee report to, in the organization.

```
`ts
//Initialize data source...
let data: object[] = [{Name: "Elizabeth", Role: "Director" },
{ Name: "Christina", ReportingPerson: "Elizabeth", Role: "Manager" },
{ Name: "Yoshi", ReportingPerson: "Christina", Role: "Lead" },
{ Name: "Philip", ReportingPerson: "Christina", Role: "Lead" },
{ Name: "Yang", ReportingPerson: "Elizabeth", Role: "Manager" },
{ Name: "Roland", ReportingPerson: "Yang", Role: "Lead" },
{ Name: "Yvonne", ReportingPerson: "Yang", Role: "Lead" }
];
`
```

Map data source

You can configure the above "Employee Information" with diagram, so that the nodes and connectors are automatically generated using the mapping properties. The following code example show how `dataSourceSettings` is used to map ID and parent with property name identifiers for employee information.

```
`ts
//Initialize data source...
let data: object[] = [{Name: "Elizabeth", Role: "Director" },
{ Name: "Christina", ReportingPerson: "Elizabeth", Role: "Manager" },
{ Name: "Yoshi", ReportingPerson: "Christina", Role: "Lead" },
{ Name: "Philip", ReportingPerson: "Christina", Role: "Lead" },
{ Name: "Yang", ReportingPerson: "Elizabeth", Role: "Manager" },
{ Name: "Roland", ReportingPerson: "Yang", Role: "Lead" },
{ Name: "Yvonne", ReportingPerson: "Yang", Role: "Lead" }
];

let items: DataManager = new DataManager(data as JSON[], new Query().take(7));
//Initialize data source...
let diagram: Diagram = new Diagram({
width: '100%', height: '600px',
//Configure data source for diagram
```

```

dataSourceSettings: {
  id: 'Name', parentId: 'ReportingPerson', dataManager: items
}
});
`

```

Visualize employee

The following code examples indicate how to define the default appearance of nodes and connectors. The `setNodeTemplate` is used to update each node based on employee data.

INDEX.JS

```

ej.diagrams.Diagram.Inject(ej.diagrams.DataBinding,
ej.diagrams.HierarchicalTree);

var data = [
  { Name: "Elizabeth", Role: "Director" },
  { Name: "Christina", ReportingPerson: "Elizabeth", Role: "Manager" },
  { Name: "Yoshi", ReportingPerson: "Christina", Role: "Lead" },
  { Name: "Philip", ReportingPerson: "Christina", Role: "Lead" },
  { Name: "Yang", ReportingPerson: "Elizabeth", Role: "Manager" },
  { Name: "Roland", ReportingPerson: "Yang", Role: "Lead" },
  { Name: "Yvonne", ReportingPerson: "Yang", Role: "Lead" }
];
var codes = {
  Director: "rgb(0, 139,139)",
  Manager: "rgb(30, 30,113)",
  Lead: "rgb(0, 100,0)"
};
function getNodeTemplate(node) {
  node.annotations[0].content = node.data.Name;
  node.style.fill = codes[node.data.Role];
}

var items = new ej.data.DataManager(data, new ej.data.Query().take(7));
var diagram = new ej.diagrams.Diagram({
  width: '100%', height: '350px',
  mode: 'SVG', snapSettings: { constraints:
ej.diagrams.SnapConstraints.None },
  dataSourceSettings: {
    id: 'Name', parentId: 'ReportingPerson', dataManager: items
  },
  layout: {
    type: 'OrganizationalChart',
    margin: { left: 10, top: 10 },
    horizontalSpacing: 50,
    verticalSpacing: 50,
    orientation: 'TopToBottom',
    getLayoutInfo: function (node, tree) {
      if (!tree.hasSubTree) {
        tree.orientation = 'Vertical';
        tree.type = 'Alternate';
      }
    }
  }
});

```

```

    },
    getNodeDefaults: function (obj, diagram) {
        obj.height = 30;
        obj.width = 70;
        obj.shape = { type: 'Basic', shape: 'Rectangle' };
        obj.annotations = [{
            id: "label1",
            style: {
                fontSize: 11,
                bold: true,
                fontFamily: "Segoe UI",
                color: "white"
            }
        }];
        return obj;
    },
    getConnectorDefaults: function (connector, diagram) {
        connector.targetDecorator.shape = 'Arrow';
        connector.type = 'Orthogonal';
        return connector;
    },
    setNodeTemplate: function (node) {
        return getNodeTemplate(node);
    }
});
diagram.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>

```

```

</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
</body>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Shapes in ##Platform_Name## Diagram control

Diagram provides support to add different kind of nodes. They are as follows:

- Text node
- Image node
- HTML node
- Native node
- Basic shapes
- Flow shapes

<!-- markdownlint-disable MD033 -->

<!-- markdownlint-disable MD010 -->

Text

Texts can be added to the diagram as [text](#) nodes. The shape property of the node allows you to set the type of node and for text nodes, it should be set as **text**. In addition, define the content object that is used to define the text to be added and style is used to customize the appearance of that text. The following code illustrates how to create a text node.

INDEX.JS

```

var node = {
    // Position of the node
    offsetX: 250,
    offsetY: 250,
    // Size of the node
    width: 100,
    height: 100,
    //Sets type of the node
    shape: { type: 'Text', content: 'Text Element' },
    //Customizes the appearances such as text, font, fill, and stroke.
    style: { strokeColor: 'none', fill: 'none', color: 'black', textAlign:
'Center' }
};
// initialize Diagram component
var diagram = new ej.diagrams.Diagram({
    width: '100%', height: '600px', nodes: [node]
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <script id="nodetemplate" type="text/x-template">
    <input type="button" id="button" value="{id}">
  </script>

  <div id="container">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Image

Diagram allows to add images as [image](#) nodes. The shape property of node allows you to set the type of node and for image nodes, it should be set as **image**. In addition, the source property of shape enables you to set the image source.

The following code illustrates how an image node is created.

INDEX.JS

```
var node = {
```



```

// Position of the node
offsetX: 250,
offsetY: 250,
// Size of the node
width: 100,
height: 100,
shape: { type: 'Image', source:
'https://ej2.syncfusion.com/demos/src/diagram/employees/image16.png' },
//Customizes the appearances such as text, font, fill, and stroke.
style: { fill: 'none' }
};
// initialize Diagram component
var diagram = new ej.diagrams.Diagram({
width: '100%', height: '600px', nodes: [node]
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <script id="nodetemplate" type="text/x-template">
    <input type="button" id="button" value="{id}">
  </script>

  <div id="container">
    <div id="element"></div>
  </div>
</script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";

```

```
}
    </script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Base64 Encoded Image Into The Image Node:

The following code illustrates how add Base64 image into image node.

INDEX.JS

[illegible]

```
// initialize Diagram component
var diagram = new ej.diagrams.Diagram({
  width: '100%', height: '600px', nodes: [node]
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <script id="nodetemplate" type="text/x-template">
    <input type="button" id="button" value="{id}">
  </script>

  <div id="container">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Note: Deploy your HTML file in the web application and export the diagram (image node) or else the image node will not be exported in the Chrome and Firefox due to security issues. Refer to the following link.

Link 1: <http://asked.online/draw-images-on-canvas-locally-using-chrome/2546077/>

Link 2: <http://stackoverflow.com/questions/4761711/local-image-in-canvas-in-chrome>

Image alignment

Stretch and align the image content anywhere but within the node boundary.

The scale property of the node allows to stretch the image as you desired (either to maintain proportion or to stretch). By default, the [scale](#) property of the node is set as **meet**.

The following code illustrates how to scale or stretch the content of the image node.

INDEX.JS

```
var node = {
  // Position of the node
  offsetX: 250,
  offsetY: 250,
  // Size of the node
  width: 100,
  height: 100,
  shape: { type: 'Image', source:
'https://ej2.syncfusion.com/demos/src/diagram/employees/image16.png',
scale: 'None' },
  //Customizes the appearances such as text, font, fill, and stroke.
  style: { fill: 'none' }
};
// initialize Diagram component
var diagram = new ej.diagrams.Diagram({
  width: '100%', height: '600px', nodes: [node]
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
```

```

</head>
<body>
  <script id="nodetemplate" type="text/x-template">
    <input type="button" id="button" value="{id}">
  </script>

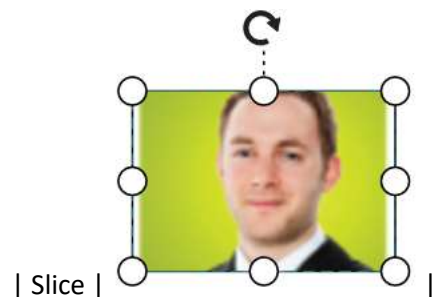
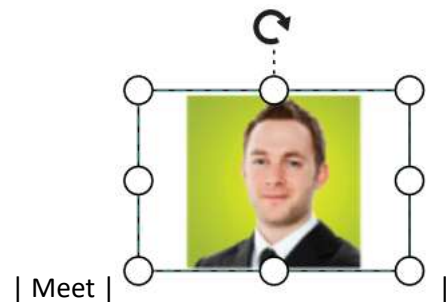
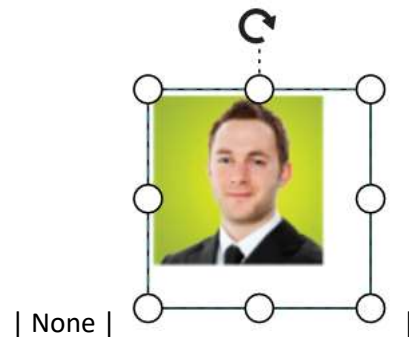
  <div id="container">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

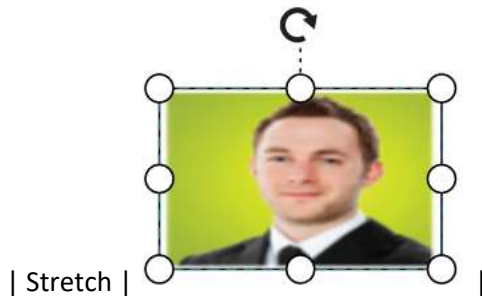
```

The following table illustrates all the possible scale options for the image node.

| Values | Images |

|-----| -----|





HTML

Html elements can be embedded in the diagram through [Html](#) type node. The shape property of node allows you to set the type of node and to create a HTML node it should be set as **HTML**. The following code illustrates how an Html node is created.

INDEX.JS

```
var node = {
  // Position of the node
  offsetX: 250,
  offsetY: 250,
  // Size of the node
  width: 100,
  height: 100,
  style: { fill: '#6BA5D7', strokeColor: 'white' },
  shape: {
    type: 'HTML',
    content: '<div
style="background:#6BA5D7;height:100%;width:100%;"><button type="button"
style="width:100px"> Button</button></div>'
  }
};

// initialize Diagram component
var diagram = new ej.diagrams.Diagram({
  width: '100%', height: '600px', nodes: [node]
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <script id="nodetemplate" type="text/x-template">
    <input type="button" id="button" value="${id}">
  </script>

  <div id="container">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Note: HTML node cannot be exported to image format, like JPEG, PNG, and BMP. It is by design, while exporting the diagram is drawn in a canvas. Further, this canvas is exported into image formats. Currently, drawing in a canvas equivalent from all possible HTML is not feasible. Hence, this limitation.

HTML Node With Template

Html elements can be embedded in the diagram using [Html](#) type node. The shape property of the node allows you to set the type of node. The following code shows how an Html node is created with a template.

INDEX.JS

```

var node = {
  // Id of the node
  id: "Node",
  // Position of the node
  offsetX: 250,
  offsetY: 250,
  // Size of the node
  width: 100,
  height: 100,
  shape: {
    type: 'HTML'
  }
};

// initialize Diagram component
var diagram = new ej.diagrams.Diagram({
  width: '100%', height: '600px', nodes: [node], nodeTemplate:
  '#nodetemplate'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <script id="nodetemplate" type="text/x-template">
    <input type="button" id="button" value="{id}">
  </script>

  <div id="container">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Native

Diagram provides support to embed SVG element into a node. The shape property of node allows you to set the type of node. To create a [native](#) node, it should be set as **native**. The following code illustrates how a native node is created.

INDEX.JS

```

var node = {
  // Position of the node

```



```

offsetX: 250,
offsetY: 250,
// Size of the node
width: 100,
height: 100,
shape: {
  type: 'Native',
  content: '<g xmlns="http://www.w3.org/2000/svg"> <g
transform="translate(1 1)"><g>   <path style="fill:#61443C;"
d="M61.979,435.057c2.645-0.512,5.291-0.853,7.936-1.109c-2.01,1.33-4.472,
1.791-6.827,1.28 C62.726,435.13,62.354,435.072,61.979,435.057z"/><path
style="fill:#61443C;"d="M502.469,502.471h-25.6c0.163-30.757-20.173-57.861-
49.749-66.304 c-5.784-1.581-11.753-2.385-17.749-2.389c-2.425-0.028-
4.849,0.114-7.253,0.427c1.831-7.63,2.747-15.45,2.731-23.296 c0.377-47.729-
34.52-88.418-81.749-95.317c4.274-0.545,8.577-0.83,12.885-
0.853c25.285,0.211,49.448,10.466,67.167,28.504
c17.719,18.039,27.539,42.382,27.297,67.666c0.017,7.846-0.9,15.666-
2.731,23.296c2.405-0.312,4.829-0.455,7.253-0.427
C472.572,434.123,502.783,464.869,502.469,502.471z"/>           </g>   <path
style="fill:#8B685A;" d="M476.869,502.471h7.536c-0.191-32.558,22.574-
60.747,54.443-67.413
c0.375,0.015,0.747,0.072,1.109,0.171c2.355,0.511,4.817,0.05,6.827-
1.28c1.707-0.085,3.413-0.171,5.12-0.171
c4.59,0,9.166,0.486,13.653,1.451c2.324,0.559,4.775,0.147,6.787-1.141c2.013-
1.288,3.414-3.341,3.879-5.685 c7.68-39.706,39.605-70.228,79.616-
76.117c4.325-0.616,8.687-0.929,13.056-0.939c13.281-
0.016,26.409,2.837,38.485,8.363
c3.917,1.823,7.708,3.904,11.349,6.229c2.039,1.304,4.527,1.705,6.872,1.106c2.
345-0.598,4.337-2.142,5.502-4.264 c14.373-25.502,39.733-42.923,68.693-
47.189h0.171c47.229,6.899,82.127,47.588,81.749,95.317c0.017,7.846-
0.9,15.666-2.731,23.296 c2.405-0.312,4.829-0.455,7.253-
0.427c5.996,0.005,11.965,0.808,17.749,2.389C456.696,444.61,477.033,471.713,4
76.869,502.471 L476.869,502.471z"/>           <path style="fill:#66993E;"
d="M502.469,7.537c0,0-6.997,264.96-192.512,252.245c-20.217-1.549-40.166-
5.59-59.392-12.032 c-1.365-0.341-2.731-0.853-4.096-1.28c0,0-0.597-2.219-
1.451-6.144c-6.656-34.048-25.088-198.997,231.765-230.144
C485.061,9.159,493.595,8.22,502.469,7.537z"/>           <path
style="fill:#9ACA5C;" d="M476.784,10.183c-1.28,26.197-16.213,238.165-
166.827,249.6 c-20.217-1.549-40.166-5.59-59.392-12.032c-1.365-0.341-
2.731-0.853-4.096-1.28c0,0-0.597-2.219-1.451-6.144
C238.363,206.279,219.931,41.329,476.784,10.183z"/>           <path
style="fill:#66993E;" d="M206.192,246.727c-0.768,3.925-1.365,6.144-
1.365,6.144c-1.365,0.427-2.731,0.939-4.096,1.28 c-21.505,7.427-
44.293,10.417-66.987,8.789C21.104,252.103,8.816,94.236,7.621,71.452c-0.085-
1.792-0.085-2.731-0.085-2.731
C222.747,86.129,211.653,216.689,206.192,246.727z"/>           <path
style="fill:#9ACA5C;" d="M180.336,246.727c-0.768,3.925-1.365,6.144-
1.365,6.144c-1.365,0.427-2.731,0.939-4.096,1.28 c-13.351,4.412-
27.142,7.359-41.131,8.789C21.104,252.103,8.816,94.236,7.621,71.452
C195.952,96.881,185.541,217.969,180.336,246.727z"/> </g>   <g>
<path d="M162.136,426.671c3.451-0.001,6.562-2.08,7.882-5.268s0.591-
6.858-1.849-9.2981-8.533-8.533 c-3.341-3.281-8.701-3.256-12.012,0.054c-
3.311,3.311-3.335,8.671-0.054,12.01218.533,8.533
C157.701,425.773,159.872,426.673,162.136,426.671L162.136,426.671z"/>
<path d="M292.636,398.57c3.341,3.281,8.701,3.256,12.012-0.054c3.311-
3.311,3.335-8.671,0.054-12.0121-8.533-8.533 c-3.341-3.281-8.701-3.256-
12.012,0.054s-3.335,8.671-0.054,12.012L292.636,398.57z"/>           </path

```

```
d="M296.169,454.771c-3.341-3.281-8.701-3.256-12.012,0.054c-3.311,3.311-3.335,8.671-0.054,12.012l8.533,8.533c3.341,3.281,8.701,3.256,12.012-0.054c3.311-3.311,3.335-8.671,0.054-12.012L296.169,454.771z"/>
    <path d="M386.503,475.37c3.341,3.281,8.701,3.256,12.012-0.054c3.311-3.311,3.335-8.671,0.054-12.012l8.533-8.533c-3.341-3.281-8.701-3.256-12.012,0.054c-3.311,3.311-3.335,8.671-0.054,12.012L386.503,475.37z"/>
    <path d="M204.803,409.604c2.264,0.003,4.435-0.897,6.033-2.518.533-8.533c3.281-3.341,3.256-8.701-0.054-12.012c-3.311-3.311-8.671-3.335-12.012-0.054l-8.533,8.533c-2.44,2.44-3.169,6.11-1.849,9.298C198.241,407.524,201.352,409.603,204.803,409.604z"/>
    <path d="M332.803,443.737c2.264,0.003,4.435-0.897,6.033-2.518.533-8.533c3.281-3.341,3.256-8.701-0.054-12.012c-3.311-3.311-8.671-3.335-12.012-0.054l-8.533,8.533c-2.44,2.44-3.169,6.11-1.849,9.298C326.241,441.658,329.352,443.737,332.803,443.737z"/>
    <path d="M341.336,366.937c2.264,0.003,4.435-0.897,6.033-2.518.533-8.533c3.281-3.341,3.256-8.701-0.054-12.012c-3.311-3.311-8.671-3.335-12.012-0.054l-8.533,8.533c-2.44,2.44-3.169,6.11-1.849,9.298C334.774,364.858,337.885,366.937,341.336,366.937z"/>
    <path d="M164.636,454.771l-8.533,8.533c-2.188,2.149-3.055,5.307-2.27,8.271c0.785,2.965,3.1,5.28,6.065,6.065c2.965,0.785,6.122-0.082,8.271-2.27l8.533-8.533c3.281-3.341,3.256-8.701-0.054-12.012C173.337,451.515,167.977,451.49,164.636,454.771L164.636,454.771z"/>
    <path d="M232.903,429.171l-8.533,8.533c-2.188,2.149-3.055,5.307-2.27,8.271c0.785,2.965,3.1,5.28,6.065,6.065c2.965,0.785,6.122-0.082,8.271-2.27l8.533-8.533c3.281-3.341,3.256-8.701-0.054-12.012C241.604,425.915,236.243,425.89,232.903,429.171L232.903,429.171z"/>
    <path d="M384.003,409.604c2.264,0.003,4.435-0.897,6.033-2.518.533-8.533c3.281-3.341,3.256-8.701-0.054-12.012c-3.311-3.311-8.671-3.335-12.012-0.054l-8.533,8.533c-2.44,2.44-3.169,6.11-1.849,9.298C377.441,407.524,380.552,409.603,384.003,409.604z"/>
    <path d="M70.77,463.304l-8.533,8.533c-2.188,2.149-3.055,5.307-2.27,8.271s3.1,5.28,6.065,6.065c2.965,0.785,6.122-0.082,8.271-2.27l8.533-8.533c3.281-3.341,3.256-8.701-0.054-12.012C79.47,460.048,74.11,460.024,70.77,463.304L70.77,463.304z"/>
    <path d="M121.97,446.238l-8.533,8.533c-2.188,2.149-3.055,5.307-2.27,8.271c0.785,2.965,3.1,5.28,6.065,6.065c2.965,0.785,6.122-0.082,8.271-2.27l8.533-8.533c3.281-3.341,3.256-8.701-0.054-12.012C130.67,442.981,125.31,442.957,121.97,446.238L121.97,446.238z"/>
    <path d="M202.302,420.638c-1.6-1.601-3.77-2.5-6.033-2.5c-2.263,0-4.433,0.899-6.033,2.51-8.533,8.533c-2.178,2.151-3.037,5.304-2.251,8.262c0.786,2.958,3.097,5.269,6.055,6.055c2.958,0.786,6.111-0.073,8.262-2.251l8.533-8.533c1.601-1.6,2.5-3.77,2.5-6.033C204.802,424.408,203.903,422.237,202.302,420.638L202.302,420.638z"/>
    <path d="M210.836,463.304c-3.341-3.281-8.701-3.256-12.012,0.054c-3.311,3.311-3.335,8.671-0.054,12.012l8.533,8.533c2.149,2.188,5.307,3.055,8.271,2.27c2.965-0.785,5.28-3.1,6.065-6.065c0.785-2.965-0.082-6.122-2.27-8.271L210.836,463.304z"/>
    <path d="M343.836,454.771l-8.533,8.533c-2.188,2.149-3.055,5.307-2.27,8.271c0.785,2.965,3.1,5.28,6.065,6.065c2.965,0.785,6.122-0.082,8.271-2.27l8.533-8.533c3.281-3.341,3.256-8.701-0.054-12.012C352.537,451.515,347.177,451.49,343.836,454.771L343.836,454.771z"/>
    <path d="M429.17,483.904c3.341,3.281,8.701,3.256,12.012-0.054s3.335-8.671,0.054-12.012l8.533-8.533c-3.341-3.281-8.701-3.256-12.012,0.054c-3.311,3.311-3.335,8.671-0.054,12.012L429.17,483.904z"/>
    <path d="M341.336,401.071c2.264,0.003,4.435-0.897,6.033-2.518.533-8.533c3.281-3.341,3.256-8.701-0.054-12.012s-8.671-3.335-12.012-0.054l-8.533,8.533c-2.44,2.44-3.169,6.11-
```

```

1.849,9.298C334.774,398.991,337.885,401.07,341.336,401.071z"/>
    <path d="M273.069,435.204c2.264,0.003,4.435-0.897,6.033-2.518.533-
8.533c3.281-3.341,3.256-8.701-0.054-12.012      s-8.671-3.335-12.012-0.054l-
8.533,8.533c-2.44,2.44-3.169,6.11-
1.849,9.298C266.508,433.124,269.618,435.203,273.069,435.204z"/>
    <path
d="M253.318,258.138c22.738,7.382,46.448,11.338,70.351,11.737c31.602,0.543,62
.581-8.828,88.583-26.796      c94.225-65.725,99.567-227.462,99.75-
234.317c0.059-2.421-0.91-4.754-2.667-6.421c-1.751-1.679-4.141-2.52-6.558-
2.308      C387.311,9.396,307.586,44.542,265.819,104.5c-28.443,42.151-
38.198,94.184-26.956,143.776c-3.411,8.366-6.04,17.03-7.852,25.881      c-
4.581-7.691-9.996-14.854-16.147-21.358c8.023-38.158,0.241-77.939-21.57-
110.261C160.753,95.829,98.828,68.458,9.228,61.196      c-2.417-0.214-
4.808,0.628-6.558,2.308c-1.757,1.667-2.726,4-
2.667,6.421c0.142,5.321,4.292,130.929,77.717,182.142
c20.358,14.081,44.617,21.428,69.367,21.008c18.624-0.309,37.097-3.388,54.814-
9.138c11.69,12.508,20.523,27.407,25.889,43.665
c0.149,15.133,2.158,30.19,5.982,44.832c-12.842-5.666-26.723-8.595-40.759-
8.6c-49.449,0.497-91.788,35.567-101.483,84.058      c-5.094-1.093-10.29-1.641-
15.5-1.638c-42.295,0.38-76.303,34.921-76.025,77.217c-
0.001,2.263,0.898,4.434,2.499,6.035
c1.6,1.6,3.771,2.499,6.035,2.499h494.933c2.263,0.001,4.434-0.898,6.035-
2.499c1.6-1.6,2.499-3.771,2.499-6.035      c0.249-41.103-31.914-75.112-72.967-
77.154c0.65-4.78,0.975-9.598,0.975-14.421c0.914-45.674-28.469-86.455-72.083-
100.045      c-43.615-13.59-90.962,3.282-
116.154,41.391C242.252,322.17,242.793,288.884,253.318,258.138L253.318,258.13
8z M87.519,238.092      c-55.35-38.567-67.358-129.25-69.833-
158.996c78.8,7.921,133.092,32.454,161.458,72.992
c15.333,22.503,22.859,49.414,21.423,76.606c-23.253-35.362-77.83-105.726-
162.473-140.577c-2.82-1.165-6.048-0.736-8.466,1.125      s-3.658,4.873-
3.252,7.897c0.406,3.024,2.395,5.602,5.218,6.761c89.261,36.751,144.772,117.77
6,161.392,144.874      C150.795,260.908,115.29,257.451,87.519,238.092z
M279.969,114.046c37.6-53.788,109.708-86.113,214.408-96.138      c-2.65,35.375-
17.158,159.05-91.892,211.175c-37.438,26.116-85.311,30.57-142.305,13.433
c19.284-32.09,92.484-142.574,212.405-191.954c2.819-1.161,4.805-3.738,5.209-
6.76c0.404-3.022-0.835-6.031-3.25-7.892      c-2.415-1.861-5.64-2.292-8.459-
1.131C351.388,82.01,279.465,179.805,252.231,222.711
C248.573,184.367,258.381,145.945,279.969,114.046L279.969,114.046z
M262.694,368.017c15.097-26.883,43.468-43.587,74.3-43.746
c47.906,0.521,86.353,39.717,85.95,87.625c-0.001,7.188-0.857,14.351-
2.55,21.337c-0.67,2.763,0.08,5.677,1.999,7.774
c1.919,2.097,4.757,3.1,7.568,2.676c1.994-0.272,4.005-0.393,6.017-
0.362c29.59,0.283,54.467,22.284,58.367,51.617H17.661      c3.899-
29.333,28.777-51.334,58.367-51.617c4-
0.004,7.989,0.416,11.9,1.254c4.622,0.985,9.447,0.098,13.417-2.467      c3.858-
2.519,6.531-6.493,7.408-11.017c7.793-40.473,43.043-69.838,84.258-
70.192c16.045-0.002,31.757,4.582,45.283,13.212
c4.01,2.561,8.897,3.358,13.512,2.205C256.422,375.165,260.36,372.163,262.694,
368.017L262.694,368.017z"/>    </g></g>'
    }
};
var diagram = new ej.diagrams.Diagram({
    width: '100%',
    height: '600px',
    nodes: [node]
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <script id="nodetemplate" type="text/x-template">
    <input type="button" id="button" value="{id}">
  </script>

  <div id="container">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Note: Like HTML node, the native node also cannot be exported to image format. Fill color of native node can be overridden by the inline style or fill of the SVG element specified in the template.

SVG content alignment

Stretch and align the svg content anywhere but within the node boundary.

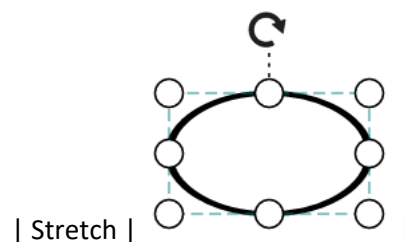
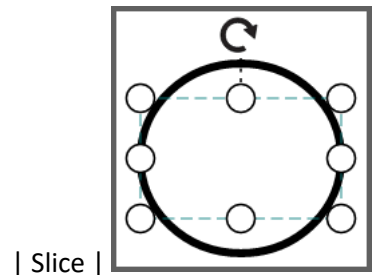
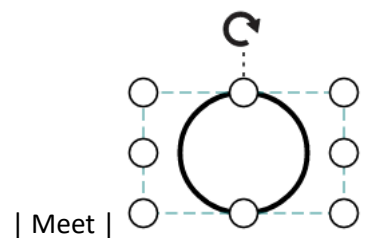
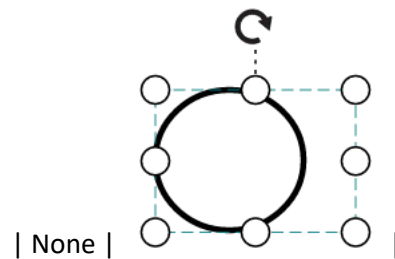
The scale property of the node allows to stretch the svg content as you desired (either to maintain proportion or to stretch). By default, the **scale** property of node is set as **meet**.

The following code illustrates how to scale or stretch the content of the node.

The following tables illustrates all the possible scale options for the node.

| Values | Images |

| ----- | ----- |



Basic shapes

- The [Basic](#) shapes are common shapes that are used to represent the geometrical information visually. To create basic shapes, the type of the shape should be set as **basic**. Its shape property can be set with any one of the built-in shape.
- To render a rounded rectangle, you need to set the type as basic and shape as rectangle. Set the [cornerRadius](#) property to specify the radius of rounded rectangle.

The following code example illustrates how to create a basic shape.

INDEX.JS

```
var node = {
```

```

// Position of the node
offsetX: 250,
offsetY: 250,
// Size of the node
width: 100,
height: 100,
style: { fill: '#6BA5D7', strokeColor: 'white' },
shape: { type: 'Basic', shape: 'Rectangle', cornerRadius: 10 },
// Text(label) added to the node
};
// initialize Diagram component
var diagram = new ej.diagrams.Diagram({
  width: '100%', height: '600px'
}, '#element');
diagram.add(node);

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <script id="nodetemplate" type="text/x-template">
    <input type="button" id="button" value="{id}">
  </script>

  <div id="container">
    <div id="element"></div>
  </div>
</script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";

```

```

}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Note: By default, the **shape** property of the node is set as **basic**.

Default property for shape is Rectangle.

Note: When the **shape** is not set for a basic shape, it is considered as a **rectangle**.

The list of basic shapes are as follows.



Path

The [Path](#) node is a commonly used basic shape that allows visually to represent the geometrical information. To create a path node, specify the shape as **path**. The path property of node allows you to define the path to be drawn. The following code illustrates how a path node is created.

INDEX.JS

```

var node = {
  // Position of the node
  offsetX: 250,
  offsetY: 250,
  // Size of the node
  width: 100,
  height: 100,
  shape: { type: 'Path', data: 'M35.2441,25 L22.7161,49.9937
L22.7161,0.00657536 L35.2441,25 z M22.7167,25 L-0.00131226,25
M35.2441,49.6337 L35.2441,0.368951 M35.2441,25 L49.9981,25'},
};
// initialize Diagram component
var diagram = new ej.diagrams.Diagram({
  width: '100%', height: '600px', nodes: [node]
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <script id="nodetemplate" type="text/x-template">
    <input type="button" id="button" value="{id}">
  </script>

  <div id="container">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Flow Shapes

The [flow](#) shapes are used to represent the process flow. It is used for analyzing, designing, and managing for documentation process. To create a flow shape, specify the shape type as **flow**. Flow shapes and by default, it is considered as **process**. The following code example illustrates how to create a flow shape.

INDEX.JS

```

var node = {
  // Position of the node
  offsetX: 250,
  offsetY: 250,
  // Size of the node
  width: 100,
  height: 100,
  shape: { type: 'Flow', shape: 'Document' },
  style: { fill: '#6BA5D7', strokeColor: 'white' },
};
// initialize Diagram component
var diagram = new ej.diagrams.Diagram({
  width: '100%', height: '600px', nodes: [node]
}, '#element');

```



```
diagram.select([diagram.nodes[0]]);
```

INDEX.HTML

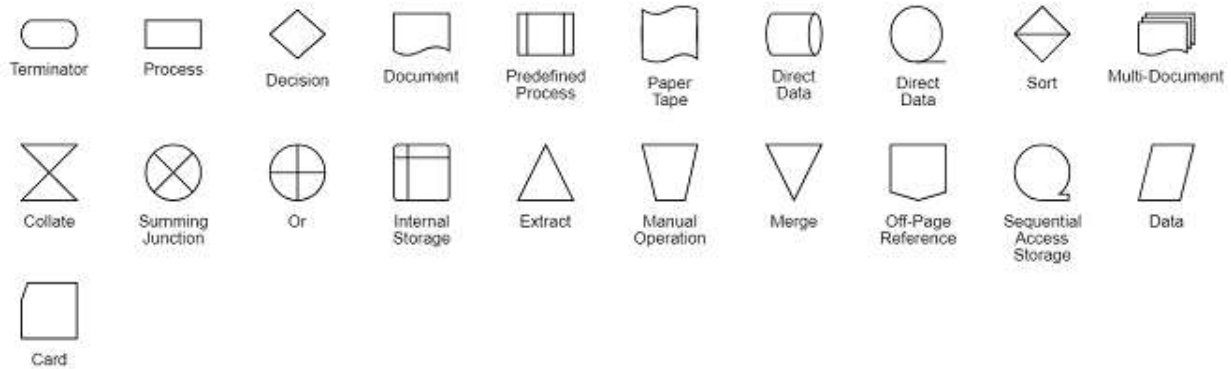
```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <script id="nodetemplate" type="text/x-template">
    <input type="button" id="button" value="{id}">
  </script>

  <div id="container">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

The list of flow shapes are as follows.



Bpmn shapes in ##Platform_Name## Diagram control

BPMN shapes are used to represent the internal business procedure in a graphical notation and enable you to communicate the procedures in a standard manner. To create a BPMN shape, in the node property shape, type should be set as “bpmn” and its shape should be set as any one of the built-in shapes. The following code example illustrates how to create a simple business process.

Note: If you want to use BPMN shapes in diagram, you need to inject BpmnDiagrams in the diagram.

INDEX.JS

```
var node = {
  // Position of the node
  offsetX: 250,
  offsetY: 250,
  // Size of the node
  width: 100,
  height: 100,
  shape: {
    type: 'Bpmn', shape: 'Event',
    event: { event: 'End' }
  },
};
// initialize Diagram component
var diagram = new ej.diagrams.Diagram({
  width: '100%', height: '600px'
}, '#element');
diagram.add(node);
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

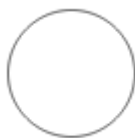
```

Note : The default value for the property **shape** is “event”.

The list of BPMN shapes are as follows:

| Shape | Image |

| ----- | ----- |



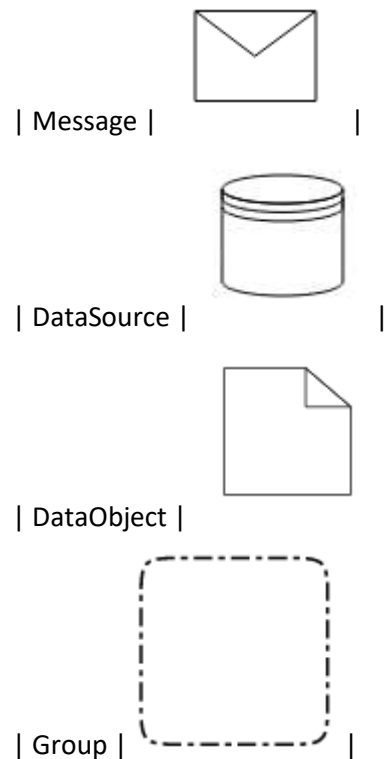
| Event |



| Gateway |



| Task |



The BPMN shapes and its types are explained as follows.

<!-- markdownlint-disable MD033 -->

Event

An [event](#) is notated with a circle and it represents an event in a business process. The type of events are as follows:

- Start
- End
- Intermediate

The [event](#) property of the node allows you to define the type of the event. The default value of the event is **start**. The following code example illustrates how to create a BPMN event.

INDEX.JS

```
var node = {  
  // Position of the node  
  offsetX: 250,  
  offsetY: 250,  
  // Size of the node  
  width: 100,  
  height: 100,  
  shape: {  
    type: 'Bpmn', shape: 'Event',  
    event: { event: 'End', trigger: 'None' }  
  },  
};  
// initialize Diagram component
```

```
var diagram = new ej.diagrams.Diagram({
  width: '100%', height: '600px'
}, '#element');
diagram.add(node);
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

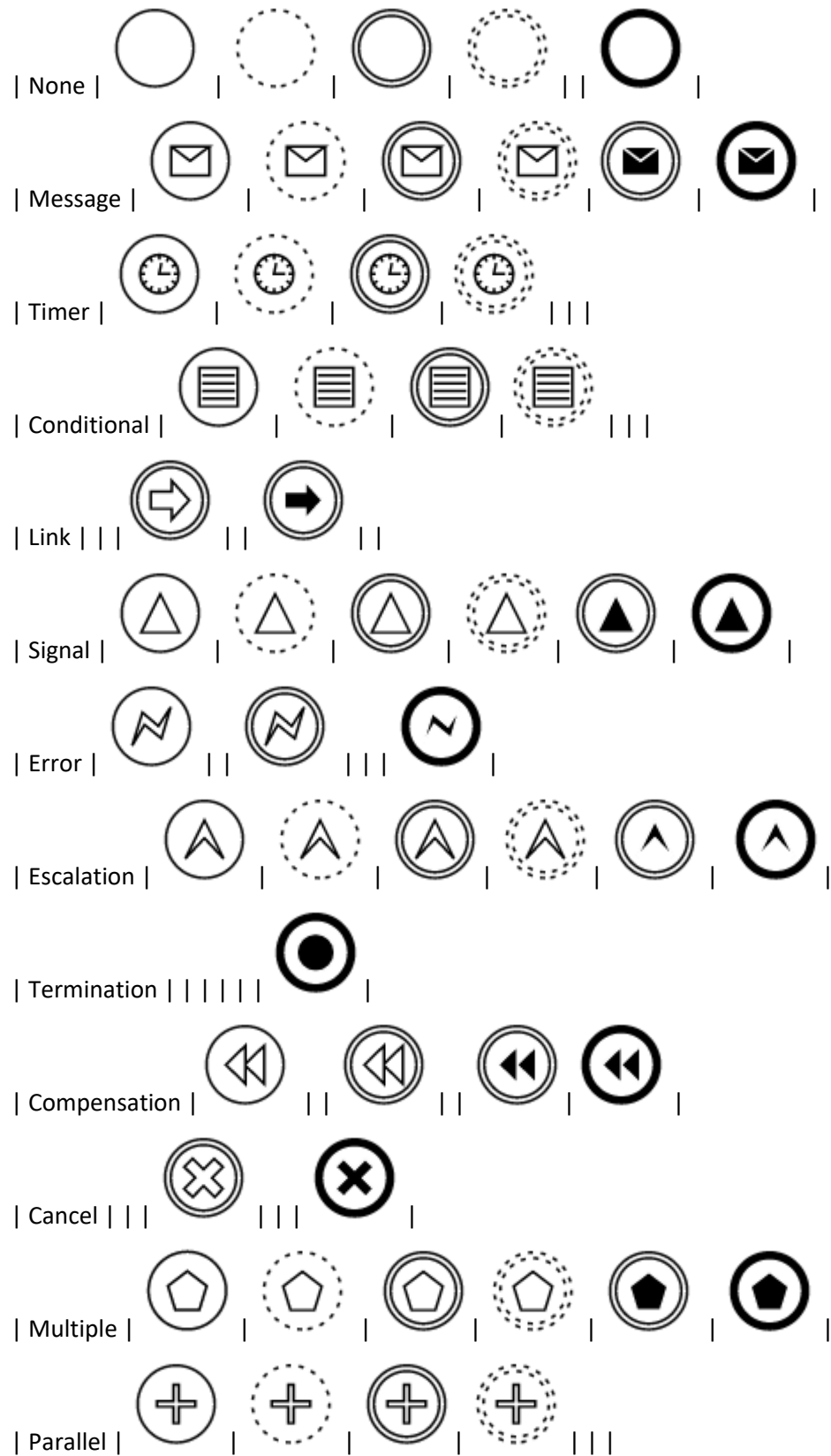
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Event triggers are notated as icons inside the circle and they represent the specific details of the process. The [trigger](#) property of the node allows you to set the type of trigger and by default, it is set as **none**. The following table illustrates the type of event triggers.

| Triggers | Start | Non-Interrupting Start | Intermediate | Non-Interrupting Intermediate | Throwing Intermediate | End |

| ----- | ----- | ----- | ----- | ----- | ----- | ----- |



Gateway

Gateway is used to control the flow of a process and it is represented as a diamond shape. To create a gateway, the shape property of the node should be set as [gateway](#) and the gateway property can be set with any of the appropriate gateways. The following code example illustrates how to create a BPMN Gateway.

INDEX.JS

```
var node = {
  // Position of the node
  offsetX: 250,
  offsetY: 250,
  // Size of the node
  width: 100,
  height: 100,
  shape: {
    type: 'Bpmn', shape: 'Gateway',
    gateway: { type: 'None' }
  },
};
// initialize Diagram component
var diagram = new ej.diagrams.Diagram({
  width: '100%', height: '600px', nodes: [node]
}, '#element');
diagram.select([diagram.nodes[0]]);
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
```







```

<div id="container">
  <div id="element"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Note: By default, the **gateway** will be set as **none**.

There are several types of gateways as tabulated:

Shape	Image
-----	-----
Exclusive	
Parallel	
Inclusive	
Complex	
EventBased	
ExclusiveEventBased	



| ParallelEventBased |

Activity

The [activity](#) is the task that is performed in a business process. It is represented by a rounded rectangle.

There are two types of activities. They are listed as follows:

- Task: Occurs within a process and it is not broken down to a finer level of detail.
- Subprocess: Occurs within a process and it is broken down to a finer level of detail.

To create a BPMN activity, set the shape as **activity**. You also need to set the type of the BPMN activity by using the activity property of the node. By default, the type of the activity is set as **task**. The following code example illustrates how to create an activity.

INDEX.JS

```
var node = {
  // Position of the node
  offsetX: 250,
  offsetY: 250,
  // Size of the node
  width: 100,
  height: 100,
  shape: {
    type: 'Bpmn', shape: 'Activity', activity: {
      activity: 'Task'
    },
  },
};

// initialize Diagram component
var diagram = new ej.diagrams.Diagram({
  width: '100%', height: '600px', nodes: [node]
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-diagrams/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The different activities of BPMN process are listed as follows.

Tasks

The [task](#) property of the node allows you to define the type of task such as sending, receiving, user based task, etc. By default, the type property of task is set as **none**. The following code illustrates how to create different types of

BPMN tasks. The [type](#) property of tasks allows to represent these results as an event attached to the task.

INDEX.JS

```

var node = {
    // Position of the node
    offsetX: 250,
    offsetY: 250,
    // Size of the node
    width: 100,
    height: 100,
    shape: {
        type: 'Bpmn', shape: 'Activity', activity: {
            activity: 'Task', task: {
                type: 'Send'
            }
        },
    },
};

// initialize Diagram component
var diagram = new ej.diagrams.Diagram({
    width: '100%', height: '600px', nodes: [node]

```

```
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

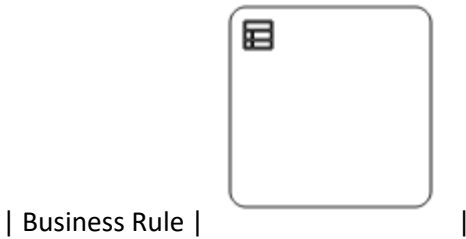
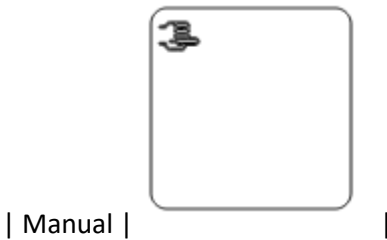
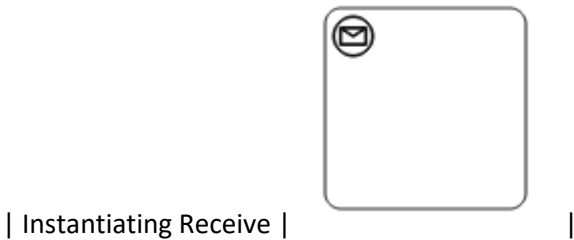
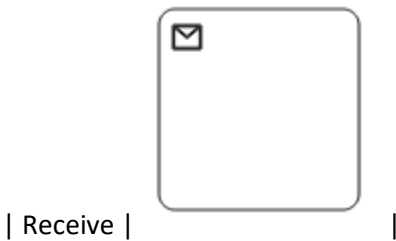
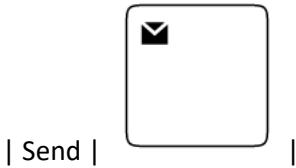
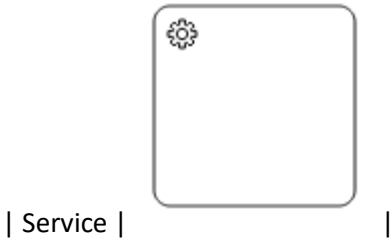
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

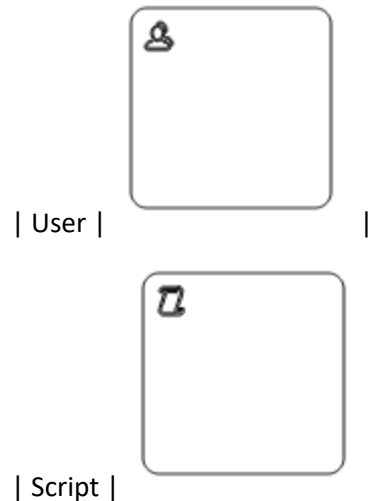
  <div id="container">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

The various types of BPMN tasks are tabulated as follows.

| Shape | Image |

| ----- | ----- |





Subprocess

A [sub-process](#) is a group of tasks, which is used to hide or reveal details of additional levels using the [collapsed](#) property.

INDEX.JS

```
var node = {
  // Position of the node
  offsetX: 250,
  offsetY: 250,
  // Size of the node
  width: 100,
  height: 100,
  shape: {
    type: 'Bpmn', shape: 'Activity', activity: {
      activity: 'SubProcess',
      subprocess: { collapsed: true }
    }
  },
};
// initialize Diagram component
var diagram = new ej.diagrams.Diagram({
  width: '100%', height: '600px', nodes: [node]
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The different types of subprocess are as follows:

- Event subprocess
- Transaction

Event subprocess

A subprocess is defined as an event subprocess, when it is triggered by an event. An event subprocess is placed within another subprocess which is not part of the normal flow of its parent process. You can set event to a subprocess with the [event](#) and [trigger](#) property of the subprocess. The [type](#) property of subprocess allows you to define the type of subprocess whether it should be event subprocess or transaction subprocess.

INDEX.JS

```

var node = {
    // Position of the node
    offsetX: 250,
    offsetY: 250,
    // Size of the node
    width: 100,
    height: 100,
    shape: {
        type: 'Bpmn', shape: 'Activity', activity: {
            activity: 'SubProcess',
            subprocess: {
                collapsed: true, type: 'Event',

```

```

        event: { event: 'Start', trigger: 'Message' }
    },
    },
    };
// initialize Diagram component
var diagram = new ej.diagrams.Diagram({
    width: '100%', height: '600px', nodes: [node]
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Transaction subprocess

- [transaction](#) is a set of activities that logically belong together, in which all contained activities must complete their parts of the transaction; otherwise the process is undone. The execution result of a transaction is one of Successful Completion, Unsuccessful Completion (Cancel), and Hazard (Exception). The [events](#) property of subprocess allows to represent these results as an event attached to the subprocess.
- The event object allows you to define the type of event by which the subprocess will be triggered. The name of the event can be defined to identify the event at runtime.
- The event's offset property is used to set the fraction/ratio (relative to parent) that defines the position of the event shape.
- The trigger property defines the type of the event trigger.
- You can also use define ports and labels to subprocess events by using event's ports and labels properties.

INDEX.JS

```
var node = {
  // Position of the node
  offsetX: 250,
  offsetY: 250,
  // Size of the node
  width: 100,
  height: 100,
  shape: {
    type: 'Bpmn', shape: 'Activity', activity: {
      activity: 'SubProcess',
      subprocess: {
        collapsed: true, type: 'Transition',
        event: [ { event: 'Intermediate', trigger: 'Cancel', offset:
{ x: 0.25, y: 1 } },
{ event: 'Intermediate', trigger: 'Error', offset: { x:
0.25, y: 1 } }, ]
      }
    },
  },
};
// initialize Diagram component
var diagram = new ej.diagrams.Diagram({
  width: '100%', height: '600px', nodes: [node]
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Process

Processes is an array collection that defines the children values for BPMN subprocess.

Loop

[Loop](#) is a task that is internally being looped. The loop property of task allows you to define the type of loop. The default value for `loop` is **none**. You can define the loop property in subprocess BPMN shape as shown in the following code.

INDEX.JS

```

var node = {
    // Position of the node
    offsetX: 100,
    offsetY: 100,
    // Size of the node
    width: 100,
    height: 100,
    shape: {
        type: 'Bpmn', shape: 'Activity', activity: {
            activity: 'Task', task: {
                loop: 'Standard'
            }
        },
    },
},

```

```

    };
    var node2 = {
        // Position of the node
        offsetX: 300,
        offsetY: 100,
        // Size of the node
        width: 100,
        height: 100,
        shape: {
            type: 'Bpmn', shape: 'Activity', activity: {
                activity: 'SubProcess', subProcess: { collapsed: true, loop:
'Standard' }
            },
        },
    };
    // initialize Diagram component
    var diagram = new ej.diagrams.Diagram({
        width: '100%', height: '600px', nodes: [node, node2]
    }, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Diagram</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {

```

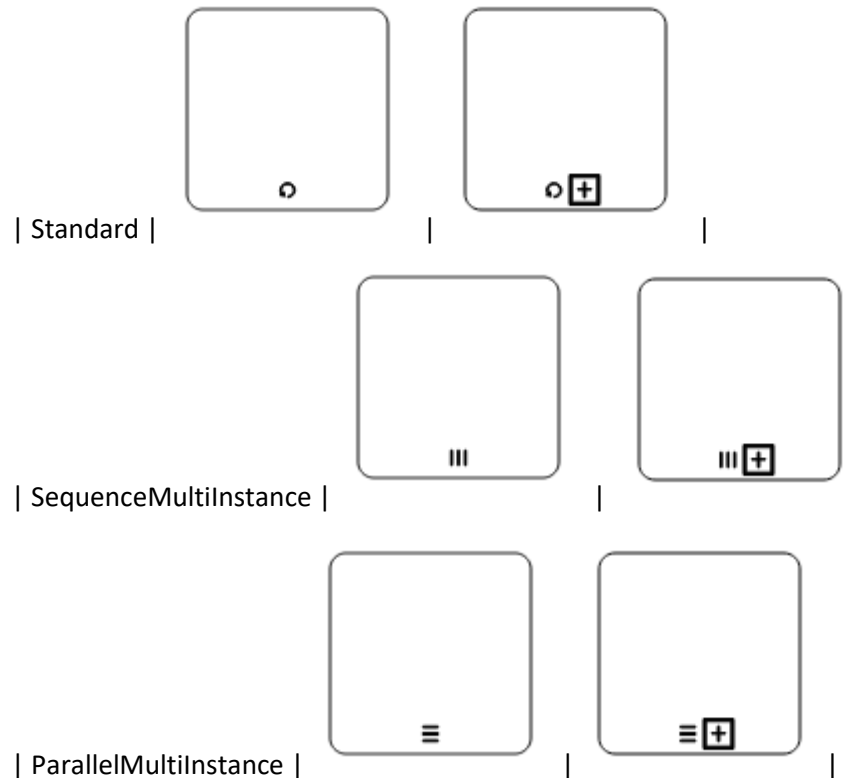
```

    ele.style.visibility = "visible";
  }
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The following table contains various types of BPMN loops.

Loops	Task	Subprocess
-----	-----	-----



Compensation

[Compensation](#) is triggered, when operation is partially failed and enabled it with the compensation property of the task and the subprocess.

INDEX.JS

```

var node = {
  // Position of the node
  offsetX: 100,
  offsetY: 100,
  // Size of the node
  width: 100,
  height: 100,
  shape: {
    type: 'Bpmn', shape: 'Activity', activity: {
      activity: 'Task', task: {
        compensation : true
      }
    }
  }
}

```

```

    },
    },
    };
var node2 = {
    // Position of the node
    offsetX: 300,
    offsetY: 100,
    // Size of the node
    width: 100,
    height: 100,
    shape: {
        type: 'Bpmn', shape: 'Activity', activity: {
            activity: 'SubProcess', subProcess: { collapsed: true,
compensation : true }
        },
    },
    };
// initialize Diagram component
var diagram = new ej.diagrams.Diagram({
    width: '100%', height: '600px', nodes: [node, node2]
}, '#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Diagram</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
</script>
```

```

var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Call

A [call](#) activity is a global subprocess that is reused at various points of the business flow and set it with the call property of the task.

INDEX.JS

```

var node = {
    // Position of the node
    offsetX: 250,
    offsetY: 250,
    // Size of the node
    width: 100,
    height: 100,
    shape: {
        type: 'Bpmn', shape: 'Activity', activity: {
            activity: 'Task', task: {
                call: true
            }
        },
    },
};
// initialize Diagram component
var diagram = new ej.diagrams.Diagram({
    width: '100%', height: '600px', nodes: [node]
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Diagram</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Adhoc

An adhoc subprocess is a group of tasks that are executed in any order or skipped in order to fulfill the end condition and set it with the [adhoc](#) property of subprocess.

INDEX.JS

```

var node = {
    // Position of the node
    offsetX: 250,
    offsetY: 250,
    // Size of the node
    width: 100,
    height: 100,
    shape: {
        type: 'Bpmn', shape: 'Activity', activity: {
            activity: 'SubProcess', subProcess: { collapsed: true, adhoc :
true }
        },
    },
    };
// initialize Diagram component
var diagram = new ej.diagrams.Diagram({
    width: '100%', height: '600px', nodes: [node]
}, '#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Diagram</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">

```

```

<link href="index.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Boundary

Boundary represents the type of task that is being processed. The [boundary](#) property of subprocess allows you to define the type of boundary. By default, it is set as **default**.

INDEX.JS

```

var node = {
    // Position of the node
    offsetX: 250,
    offsetY: 250,
    // Size of the node
    width: 100,
    height: 100,
    shape: {
        type: 'Bpmn', shape: 'Activity', activity: {
            activity: 'SubProcess', subProcess: { collapsed: true, boundary:
'Call' }
        },
    },
};

// initialize Diagram component
var diagram = new ej.diagrams.Diagram({

```

```
width: '100%', height: '600px', nodes: [node]
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

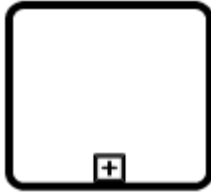
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

The following table contains various types of BPMN boundaries.

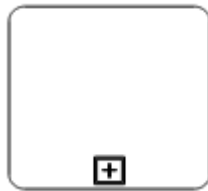
Boundary	Image
-----	-----



| Call |



| Event |



| Default |

Data

A data object represents information flowing through the process, such as data placed into the process, data resulting from the process, data that needs to be collected, or data that must be stored. To define a [data object](#), set the shape as **DataObject** and the type property defines whether data is an input or an output. You can create multiple instances of data object with the collection property of data.

INDEX.JS

```
var node = {
  // Position of the node
  offsetX: 250,
  offsetY: 250,
  // Size of the node
  width: 100,
  height: 100,
  shape: {
    type: 'Bpmn', shape: 'DataObject',
    dataObject: { collection: true, type: 'Input' }
  }
};

// initialize Diagram component
var diagram = new ej.diagrams.Diagram({
  width: '100%', height: '600px', nodes: [node]
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
```

```
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

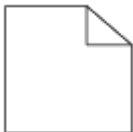
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

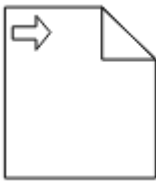
The following table contains various representation of BPMN data object.

| Boundary | Image |

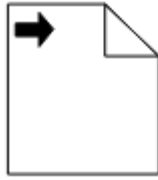
| ----- | ----- |



| Collection Data Object | |



| Data Input | |



| Data Output |

Datasource

Datasource is used to store or access data associated with a business process. To create a datasource, set the shape as **datasource**. The following code example illustrates how to create a datasource.

INDEX.JS

```
var node = {
  // Position of the node
  offsetX: 250,
  offsetY: 250,
  // Size of the node
  width: 100,
  height: 100,
  shape: {
    type: 'Bpmn', shape: 'DataSource',
  }
};
// initialize Diagram component
var diagram = new ej.diagrams.Diagram({
  width: '100%', height: '600px', nodes: [node]
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
```

```

</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
</body>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Artifact

Artifact is used to show additional information about a process in order to make it easier to understand. There are two types of artifacts in BPMN.

- Text annotation
- Group

Text annotation

- A BPMN object can be associated with a text annotation which does not affect the flow but gives details about objects within a flow. The annotation property of the node is used to connect an annotation element to the BPMN node.
- The annotation element can be displaced into a different position interactively by dragging the annotation to a particular position.
- The annotation element can be switched from a BPMN node to another BPMN node simply by dragging the source end of the annotation connector into the other BPMN node.
- The annotation angle property is used to set the angle between the BPMN shape and the annotation.
- The annotation direction property is used to set the direction of the text annotation.
- To set the size for text annotation, use width and height properties.
- The annotation length property is used to set the distance between the BPMN shape and the annotation.
- The annotation text property defines the additional information about the flow object in a BPMN process.

INDEX.JS

```

var node = {
    // Position of the node
    offsetX: 100,
    offsetY: 100,
    // Size of the node
    width: 100,
    height: 100,
    shape: {
        type: 'Bpmn',

```

```

    shape: 'DataObject',
    dataObject: {
      collection: true,
      type: 'Input'
    },
    annotations: [{
      id: 'left',
      angle: 45,
      length: 150,
      text: 'Left',
    }]
  }
};
// initialize Diagram component
var diagram = new ej.diagrams.Diagram({
  width: '100%', height: '600px', nodes: [node]
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}

```

```

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Group

A group is used to frame a part of the diagram, shows that elements included in it are logically belong together and does not have any other semantics other than organizing elements. To create a group, the shape property of the node should be set as **group**. The following code example illustrates how to create a BPMN group.

INDEX.JS

```

var node = {
  // Position of the node
  offsetX: 250,
  offsetY: 250,
  // Size of the node
  width: 100,
  height: 100,
  shape: {
    type: 'Bpmn', shape: 'Group',
  }
};
// initialize Diagram component
var diagram = new ej.diagrams.Diagram({
  width: '100%', height: '600px', nodes: [node]
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>

```

```

</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

BPMN flows

[BPMN Flows](#) are lines that connects BPMN flow objects.

Association

[BPMN Association](#) flow is used to link flow objects with its corresponding text or artifact. An association is represented as a dotted graphical line with opened arrow. The types of association are as follows:

- Directional
- BiDirectional
- Default

The `association` property allows you to define the type of association. The following code example illustrates how to create an association.

INDEX.JS

```

var connector = {
    // Position of the node
    sourcePoint: { x: 100, y: 200 }, targetPoint: { x: 300, y: 200 }, type:
    'Orthogonal',
    shape: {
type: 'Bpmn',
flow: 'Association',
association: 'BiDirectional'
    },
    };
// initialize Diagram component
var diagram = new ej.diagrams.Diagram({
    width: '100%', height: '600px', connectors: [connector]
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Diagram</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">

```

```

<link href="index.css" rel="stylesheet">




<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The following table demonstrates the visual representation of association flows.

Association	Image
Default	
Directional	
BiDirectional	

Note : The default value for the property **association** is **default**.

Sequence

A [sequence](#) flow shows the order in which the activities are performed in a BPMN process and is represented by a solid graphical line. The types of sequence are as follows:

- Normal
- Conditional
- Default

The sequence property allows you to define the type of sequence. The following code example illustrates how to create a sequence flow.

INDEX.JS

```
var connector = {
  // Position of the node
  sourcePoint: { x: 100, y: 200 }, targetPoint: { x: 300, y: 200 }, type:
  'Orthogonal',
  shape: {
    type: 'Bpmn',
    flow: 'Sequence',
    sequence: 'Conditional'
  },
};

// initialize Diagram component
var diagram = new ej.diagrams.Diagram({
  width: '100%', height: '600px', connectors: [connector]
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>


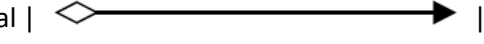
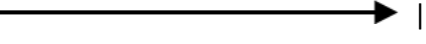
  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
```

```

    ele.style.visibility = "visible";
  }
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The following table contains various representation of sequence flows.

Sequence	Image
Default	
Conditional	
Normal	

Note: The default value for the property `sequence` is **normal**.

Message

A [message](#) flow shows the flow of messages between two participants and is represented by dashed line. The types of message are as follows:

- InitiatingMessage
- NonInitiatingMessage
- Default

The message property allows you to define the type of message. The following code example illustrates how to define a message flow.

INDEX.JS

```

var connector = {
  // Position of the node
  sourcePoint: { x: 100, y: 200 }, targetPoint: { x: 300, y: 200 }, type:
  'Orthogonal',
  shape: {
    type: 'Bpmn',
    flow: 'Message',
    message: 'InitiatingMessage'
  },
};
// initialize Diagram component
var diagram = new ej.diagrams.Diagram({
  width: '100%', height: '600px', connectors: [connector]
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">

```

```

<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

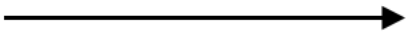


<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The following table contains various representation of message flows.

Message	Image
-----	-----
Default	
InitiatingMessage	
NonInitiatingMessage	

Note: The default value for the property `message` is **default**.

UML diagram in ##Platform_Name## Diagram control

UML Class Diagram

A class diagram visually depicts the static structure of an application and is extensively employed in modeling object-oriented systems. It holds a unique position in UML diagrams, as it directly aligns with object-oriented languages. The diagram also facilitates the automatic generation of class diagram

shapes based on business logic, streamlining the translation from conceptual models to practical implementation.

UML Class Diagram Shapes

The UML class diagram shapes are explained as follows.

Class

- A class defines a group of objects that share common specifications, features, constraints, and semantics. To create a class object, the classifier should be defined using the [class] ([../api/diagram/umlClassifierShapeModel#class](#)) notation. This notation serves as a foundational element in object-oriented programming, encapsulating the essential characteristics and behavior that objects belonging to the class will exhibit.
- Also, define the [name](#), [attributes](#), and [methods](#) of the class using the class property of node.
- The attribute's [name](#), [type](#), and [scope](#) properties allow you to define the name, data type, and visibility of the attribute.
- The method's [name](#), [parameters](#), [type](#), and [scope](#) properties allow you to define the name, parameter, return type, and visibility of the methods.
- The method parameters object properties allow you to define the name and type of the parameter.
- The following code example illustrates how to create a class.

INDEX.JS

```
/**
 * Tooltip sample
 */
var diagram;
var nodes = [
  {
    id: "Patient",
    style: {
      fill: '#26A0DA',
    },
    shape: {
      type: "UmlClassifier",
      classShape: {
        name: "Patient",
        attributes: [
          createProperty("allergies", "String[*]")
        ],
        methods: [createMethods("getHistory", "History")]
      },
      classifier: "Class"
    },
    offsetX: 200,
    offsetY: 250
  }
];
function createProperty(name, type) {
  return { name: name, type: type };
}
function createMethods(name, type) {
  return { name: name, type: type };
}
```

```

}
diagram = new ej.diagrams.Diagram({
  width: '100%',
  height: '600px',
  nodes: nodes,
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  popups/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  diagrams/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  navigations/styles/fabric.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Interface

- An interface is a specific type of classifier that signifies a declaration of a cohesive set of public features and obligations. When creating an interface, involves defining the classifier property

using the [interface](#) notation. This essential concept in object-oriented programming outlines a contract for classes to adhere to, specifying the required methods and behaviors without delving into the implementation details.

- Also, define the [name](#), [attributes](#), and [methods](#) of the interface using the interface property of the node.
- The attribute's name, type, and scope properties allow you to define the name, data type, and visibility of the attribute.
- The method's name, parameter, type, and scope properties allow you to define the name, parameter, return type, and visibility of the methods.
- The method parameter object properties of name and type allow you to define the name and type of the parameter.
- The following code example illustrates how to create an interface.

INDEX.JS

```
/**
 * Tooltip sample
 */
var node = {
  id: 'node',
  offsetX: 400,
  offsetY: 300,
  style: {
    fill: '#26A0DA',
  },
  shape: {
    type: 'UmlClassifier',
    interfaceShape: {
      name: "Bank Account",
      property: [{
        name: "owner",
        type: "String[*]", style: {}
      },
      {
        name: "balance",
        type: "Dollars"
      }
    ],
    methods: [{
      name: "deposit", style: {},
      parameters: [{
        name: "amount",
        type: "Dollars",
        style: {}
      }
    ]
  }
],
  },
  classifier: 'Interface'
},
];
var diagram = new ej.diagrams.Diagram({
  width: '100%',
  height: '600px',
  nodes: [node],
}, '#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Enumeration

- To establish an enumeration, designate the classifier property of the node as [enumeration](#). Additionally, define the name and enumerate the members of the enumeration using the appropriate enumeration property of the node. This process encapsulates a set of distinct values within the enumeration, allowing for a clear representation of specific, and named constants within a system.
- You can set a name for the enumeration members collection using the name property of the members collection.

- The following code example illustrates how to create an enumeration.

INDEX.JS

```

/**
 * Tooltip sample
 */
var diagram;
var node = {
  id: 'node',
  offsetX: 300,
  offsetY: 200,
  style: {
    fill: '#26A0DA',
  },
  shape: {
    type: 'UmlClassifier',
    enumerationShape: {
      name: 'AccountType',
      members: [
        {
          name: 'Checking Account', style: {}
        },
        {
          name: 'Savings Account'
        },
        {
          name: 'Credit Account'
        }
      ]
    },
    classifier: 'Enumeration'
  },
};
diagram = new ej.diagrams.Diagram({
  width: '100%',
  height: '600px',
  nodes: [node],
}, '#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  popups/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  buttons/styles/material.css" rel="stylesheet">
```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

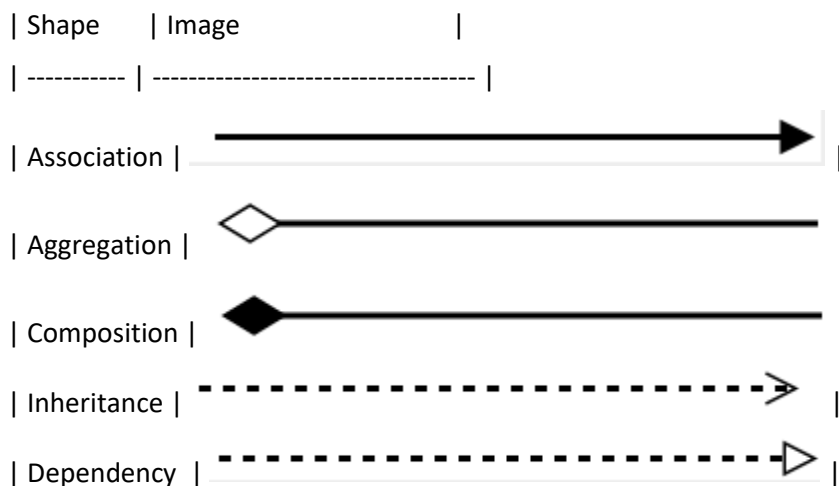
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

UML Class Relationships

- A class may be involved in one or more relationships with other classes. A relationship can be one of the following types:



Association

Association is basically a set of links that connects elements of a UML model. The type of association is as follows.

1. Directional
2. BiDirectional

The association property allows you to define the type of association. The default value of association is "Directional". The following code example illustrates how to create an association.

INDEX.JS

```
/**
 * Tooltip sample
 */
var diagram;
var connector = {
  id: "connector",
  sourcePoint: { x: 100, y: 100 },
  targetPoint: { x: 300, y: 300 },
  type: "Straight",
  shape: {
    type: "UmlClassifier",
    relationship: "Association",
    association: "BiDirectional"
  }
};
diagram = new ej.diagrams.Diagram({
  width: '650px',
  height: '350px',
  connectors: [connector],
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  popups/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  diagrams/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  navigations/styles/fabric.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Aggregation

Aggregation is a binary association between a property and one or more composite objects that group together a set of instances. Aggregation is decorated with a hollow diamond. To create an aggregation shape, define the relationship as “aggregation”.

The following code example illustrates how to create an aggregation.

INDEX.JS

```

/**
 * Tooltip sample
 */
var diagram;
var connector = {
    id: "connector",
    sourcePoint: { x: 100, y: 100 },
    targetPoint: { x: 300, y: 300 },
    type: "Straight",
    shape: {
        type: "UmlClassifier",
        relationship: "Aggregation"
    }
};
diagram = new ej.diagrams.Diagram({
    width: '650px',
    height: '350px',
    connectors: [connector],
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Diagram</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Composition

Composition is a “strong” form of “aggregation”. The composition is decorated with a black diamond. To create a composition shape, define the relationship property of the connector as “composition”.

The following code example illustrates how to create a composition.

INDEX.JS

```

/**
 * Tooltip sample
 */
var diagram;
var connector = {
    id: "connector",
    sourcePoint: { x: 100, y: 100 },
    targetPoint: { x: 300, y: 300 },
    type: "Straight",
    shape: {
        type: "UmlClassifier",
        relationship: "Composition"
    }
}

```

```
};
diagram = new ej.diagrams.Diagram({
  width: '650px',
  height: '350px',
  connectors: [connector],
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  popups/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  diagrams/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  navigations/styles/fabric.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Dependency

Dependency is a directed relationship, which is used to show that some UML elements need or depend on other model elements for specifications. Dependency is shown as a dashed line with an opened arrow. To create a dependency, define the relationship property of the connector as “dependency”.

The following code example illustrates how to create a dependency.

INDEX.JS

```
/**
 * Tooltip sample
 */
var diagram;
var connector = {
  id: "connector",
  sourcePoint: { x: 100, y: 100 },
  targetPoint: { x: 300, y: 300 },
  type: "Straight",
  shape: {
    type: "UmlClassifier",
    relationship: "Dependency"
  }
};
diagram = new ej.diagrams.Diagram({
  width: '650px',
  height: '350px',
  connectors: [connector],
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  popups/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  diagrams/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  navigations/styles/fabric.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
```

```

        <div id="element"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Inheritance

Inheritance is also called a “generalization”. Inheritance is a binary taxonomic directed relationship between a more general classifier (superclass) and a more specific classifier (subclass). Inheritance is shown as a line with a hollow triangle.

To create an inheritance, define the relationship as “inheritance”.

The following code example illustrates how to create an inheritance.

INDEX.JS

```

/**
 * Tooltip sample
 */
var diagram;
var connector = {
    id: "connector",
    sourcePoint: { x: 100, y: 100 },
    targetPoint: { x: 300, y: 300 },
    type: "Straight",
    shape: {
        type: "UmlClassifier",
        relationship: "Inheritance"
    }
};
diagram = new ej.diagrams.Diagram({
    width: '650px',
    height: '350px',
    connectors: [connector],
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Diagram</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
    popups/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
    base/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Multiplicity

Multiplicity is a definition of an inclusive interval of non-negative integers to specify the allowable number of instances of a described element. The type of multiplicity are as follows.

1. OneToOne
2. ManyToOne
3. OneToMany
4. ManyToMany
 - By default the multiplicity will be considered as “OneToOne”.
 - The multiplicity property in UML allows you to specify large number of elements or some collection of elements.
 - The shape multiplicity’s source property is used to set the source label to the connector and the target property is used to set the target label to the connector.
 - To set an optionality or cardinality for the connector source label, use the optional property.
 - The [lowerBounds](#) and [upperBounds](#) could be natural constants or constant expressions evaluated to a natural (non negative) number. The upper bound could also be specified as an asterisk ‘*’ which denotes an unlimited number of elements. The upper bound should be greater than or equal to the lower bound.
 - The following code example illustrates how to customize the multiplicity.

INDEX.JS


```

/**
 * Tooltip sample
 */
var diagram;
var connector = {
  id: "connector",
  sourcePoint: { x: 100, y: 100 },
  targetPoint: { x: 300, y: 300 },
  type: "Straight",
  shape: {
    type: "UmlClassifier",
    relationship: "Dependency",
    multiplicity: {
      type: "OneToMany",
      source: {
        optional: true,
        lowerBounds: 89,
        upperBounds: 67
      },
      target: {
        optional: true,
        lowerBounds: 78,
        upperBounds: 90
      }
    }
  }
};
diagram = new ej.diagrams.Diagram({
  width: '650px',
  height: '350px',
  connectors: [connector],
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  popups/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  diagrams/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

How to add UML child at runtime

In UML nodes, child elements such as members, methods and attributes can be added either programmatically or interactively.

Adding UML child through code

The [addChildToUmlNode](#) method is employed for dynamically adding a child to the UML node during runtime, providing flexibility in modifying the diagram structure programmatically.

The following code illustrates how to add methods to UML nodes in the diagram.

```

`ts
let node = diagram.selectedItems.nodes[0];

let methods = { name: 'getHistory', style: { color: "red", }, parameters: [{ name: 'Date', style: {} }], type:
'History' };

diagram.addChildToUmlNode(node, methods, 'Methods');
`

```

The following code illustrates how to add attributes to UML nodes in the diagram.

```

`ts
let node = diagram.selectedItems.nodes[0];

let attributes = { name: 'accepted', type: 'Date', style: { color: "red", } };

diagram.addChildToUmlNode(node, attributes, "Attributes");
`

```

The following code illustrates how to add members to UML nodes in the diagram.

```

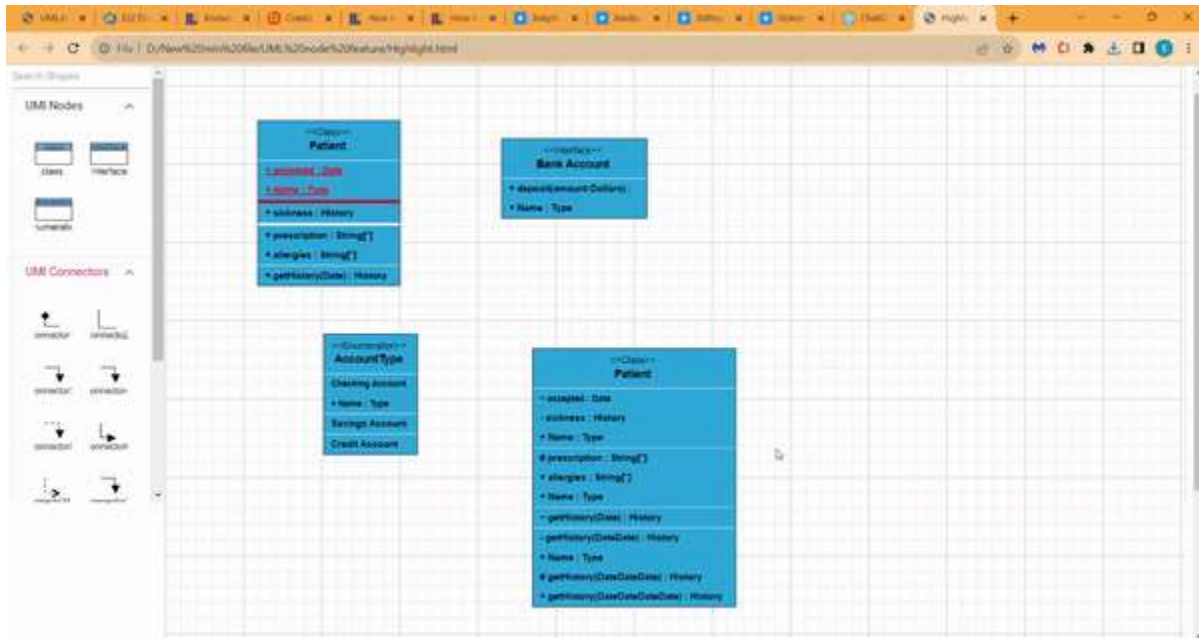
`ts

```

```
let node = diagram.selectedItems.nodes[0];
let members = { name: 'Checking new', style: { color: "red", }, isSeparator: true };
diagram.addChildToUmlNode(node, members, "Members");
`
```

Adding UML child through user interaction

To include a child, select a node, move the mouse outside it, and position the pointer near the right side. A highlighter emerges between the two child elements. Click the highlighter to add a child type to the chosen UML node seamlessly. The following gif illustrates how to add a Child through user interaction.



Adding UML Nodes in Symbol palette

UML built-in shapes are efficiently rendered in a symbol palette. The `symbols` property is utilized to define UML symbols with the necessary classes and methods. By incorporating this feature, you can seamlessly augment the palette with a curated collection of predefined UML symbols, thereby enhancing the versatility of your UML diagramming application.

The following code example showcases the rendering of UML built-in shapes in a symbol palette

INDEX.JS

```
//Initialize the flowshapes for the symbol palette
function getUmlShapes() {
  var flowShapes = [
    {
      id: 'class',
      style: {
        fill: '#26A0DA',
      },
      borderColor: 'white',
      shape: {
        type: 'UmlClassifier',
        classShape: {
          attributes: [
```

```

        { name: 'accepted', type: 'Date', style: { color:
"red", fontFamily: "Arial", textDecoration: 'Underline', italic: true
}, isSeparator: true },
        ],
        methods: [{ name: 'getHistory', style: {}, parameters:
[ { name: 'Date', style: {} } ], type: 'History' }],
        name: 'Patient'
    },
    classifier: 'Class'
},
{
    id: 'Interface',
    style: {
        fill: '#26A0DA',
    }, borderColor: 'white',
    shape: {
        type: 'UmlClassifier',
        interfaceShape: {
            name: "Bank Account",
        },
        classifier: 'Interface'
    },
},
{
    id: 'Enumeration',
    style: {
        fill: '#26A0DA',
    }, borderColor: 'white',
    shape: {
        type: 'UmlClassifier',
        enumerationShape: {
            name: 'AccountType',
            members: [
                {
                    name: 'Checking Account', style: {}
                },
            ],
        },
        classifier: 'Enumeration'
    },
},
];
return flowShapes;
}
function setPaletteNodeDefaults(node) {
    node.width = 100;
    node.height = 100;
}
var palette = new ej.diagrams.SymbolPalette({
    palettes: [
        { id: 'UML', expanded: true, symbols: getUmlShapes(), title:
'UMLClass Nodes' },
    ],
    width: '100%', height: '100%', symbolHeight: 90, symbolWidth: 90,
    getNodeDefaults: setPaletteNodeDefaults,
    //Defines the symbol description for the symbols in the palette

```

```

    getSymbolInfo: function (symbol) {
        return { fit: true, description: { text: symbol.id, }};
    }
});
palette.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  popups/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  diagrams/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  navigations/styles/fabric.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

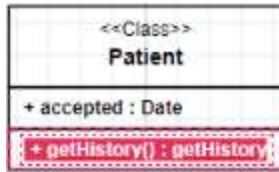
  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Editing in UML nodes

You can edit the name, attributes, and methods of the class diagram shapes just double clicking, similar to editing a node annotation.

The following image illustrates how the text editor looks in an edit mode.



UML Activity diagram

An Activity diagram functions as a visual flowchart, illustrating the progression from one activity to the next within a system. Each activity corresponds to a system operation, providing a clear depiction of the sequential flow in a dynamic process..

The purpose of an activity diagram can be described as follows.

1. Draw the activity flow of a system.
2. Describe the sequence from one activity to another.
3. Describe the parallel, branched, and concurrent flow of the system.

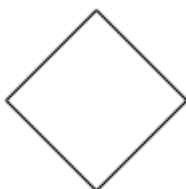
UML Activity diagram Shapes

To create a UmlActivity, define the type as "UmlActivity" and the list of built-in shapes as demonstrated as follows and it should be set in the "shape" property.

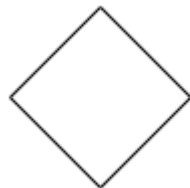
Shape	Image	
-----	-----	



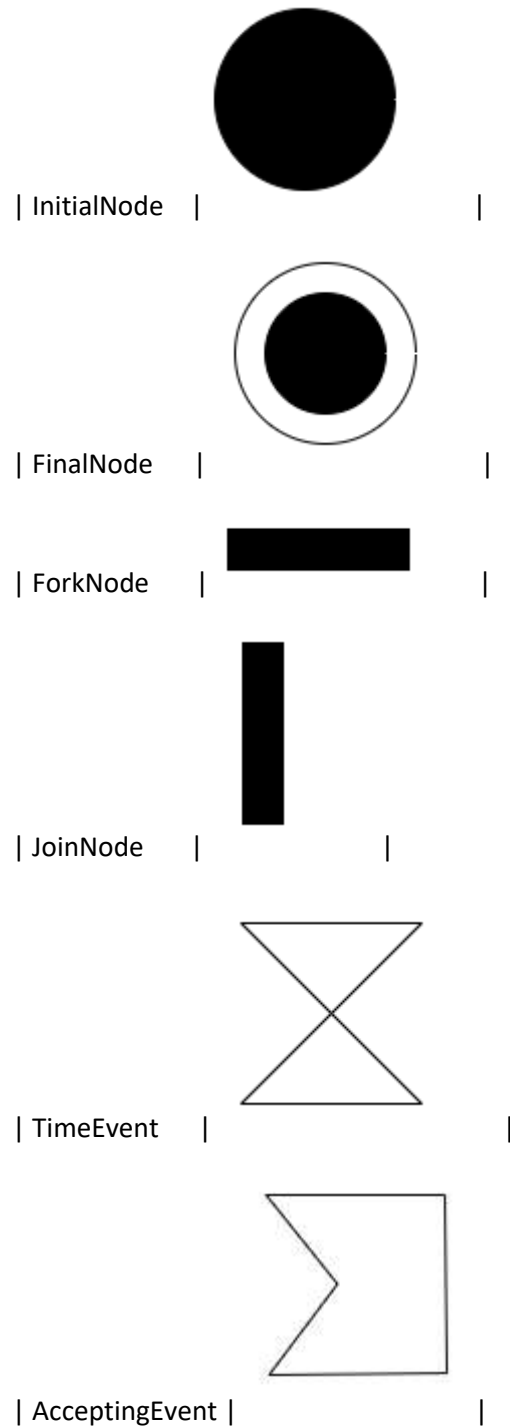
Action		
--------	--	--

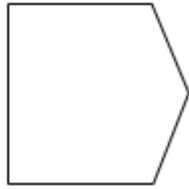


Decision		
----------	--	--

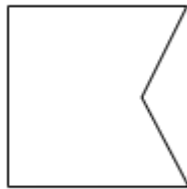


MergeNode		
-----------	--	--





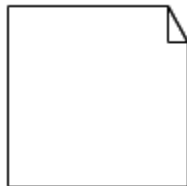
| SendSignal |



| ReceiveSignal |



| StructuredNode |



| Note |

The following code illustrates how to create a UmlActivity shapes.

INDEX.JS

```
/**
 * Tooltip sample
 */
var node = {
  id: "UmlDiagram",
  //Set node size
  width: 100,
  height: 100,
  //position the node
  offsetX: 200,
  offsetY: 200,
  shape: {
    type: "UmlActivity",
    //Define UmlActivity shape
```



```

        shape: "Action"
    }
};
var diagram = new ej.diagrams.Diagram({
    width: '100%',
    height: '600px',
    nodes: [node]
}, '#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Diagram</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

UML Activity connector

To establish a UML Activity connector, specify the type as "UmlActivity" and define the flow as either "Exception," "Control," or "Object." This configuration delineates the nature of the connection, allowing for a precise representation of the interaction within the activity diagram.

The following code illustrates how to create a UmlActivity connector.

INDEX.JS

```
/**
 * UmlActivity sample
 */
var diagram;
var connector = {
  id: 'connector',
  type: 'Straight',
  //Define connector start and end points
  sourcePoint: { x: 100, y: 700 },
  targetPoint: { x: 200, y: 800 },
  shape: { type: 'UmlActivity', flow: 'Exception' }
};
diagram = new ej.diagrams.Diagram({
  width: '100%',
  height: '600px',
  connectors: [connector],
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  popups/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  diagrams/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  navigations/styles/fabric.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
```

```

</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
</body>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Group in ##Platform_Name## Diagram control

Group is used to cluster multiple nodes and connectors into a single element. It acts like a container for its children (nodes, groups, and connectors). Every change made to the group also affects the children. Child elements can be edited individually.

Create group

Add group when initializing diagram

A group can be added to the diagram model through [nodes](#) collection. To define an object as group, add the child objects to the [children](#) collection of the group. The following code illustrates how to create a group node.

- The [padding](#) property of a group node defines the spacing between the group node's edges and its children.
- While creating group, its child node need to be declared before the group declaration.
- Add a node to the existing group child by using the `diagram.group` method.
- The group's `diagram.unGroup` method is used to define whether the group can be ungrouped or not.
- A group can be added into a child of another group.

INDEX.JS

```

var nodes = [{
    id: "rectangle1",
    offsetX: 100,
    offsetY: 100,
    width: 100,
    height: 100,
    style: {
        strokeColor: '#6BA5D7',
        fill: '#6BA5D7'
    },
    annotations: [{
        content: 'rectangle1'
    }]
}, {
    id: "rectangle2",
    offsetX: 200,

```

```

        offsetY: 200,
        width: 100,
        height: 100,
        style: {
            strokeColor: '#6BA5D7',
            fill: '#6BA5D7'
        },
        annotations: [{
            content: 'rectangle2'
        }]
    },
    {
        id: 'group',
        children: ['rectangle1', 'rectangle2'],
        padding: {left: 10, right: 10, top: 10, bottom: 10}
    },
];
var diagram = new ej.diagrams.Diagram({
    width: '1500px',
    height: '600px',
    getNodeDefaults: function(node) {
        node.height = 100;
        node.width = 100;
        node.style.fill = '#6BA5D7';
        node.style.strokeColor = 'white';
        return node;
    },
    nodes: nodes,
}, '#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The following code illustrates how a ungroup at runtime.

INDEX.JS

```

var nodes = [{
    id: "rectangle1",
    offsetX: 100,
    offsetY: 100,
    width: 100,
    height: 100,
    style: {
        strokeColor: '#6BA5D7',
        fill: '#6BA5D7'
    },
    annotations: [{
        content: 'rectangle1'
    }]
}, {
    id: "rectangle2",
    offsetX: 200,
    offsetY: 200,
    width: 100,
    height: 100,
    style: {
        strokeColor: '#6BA5D7',
        fill: '#6BA5D7'
    },
    annotations: [{
        content: 'rectangle2'
    }]
},
{
    id: 'group',
    children: ['rectangle1', 'rectangle2']
},
];
var diagram = new ej.diagrams.Diagram({
    width: '1500px',
    height: '600px',

```

```

nodes: nodes,
getNodeDefaults: function(node) {
    node.height = 100;
    node.width = 100;
    node.style.fill = '#6BA5D7';
    node.style.strokeColor = 'white';
    return node;
},
}, '#element');
diagram.selectAll();
diagram.unGroup();

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Add group at runtime

A group node can be added at runtime by using the client-side method `diagram.add`.

The following code illustrates how a group node is added at runtime.

INDEX.JS

```
var nodes = [{
  id: "rectangle1",
  offsetX: 100,
  offsetY: 100,
  width: 100,
  height: 100,
  style: {
    strokeColor: '#6BA5D7',
    fill: '#6BA5D7'
  },
  annotations: [{
    content: 'rectangle1'
  }]
}, {
  id: "rectangle2",
  offsetX: 200,
  offsetY: 200,
  width: 100,
  height: 100,
  style: {
    strokeColor: '#6BA5D7',
    fill: '#6BA5D7'
  },
  annotations: [{
    content: 'rectangle2'
  }]
},
];

var group = {
  id: 'group',
  children: ['rectangle1', 'rectangle2']
};

var diagram = new ej.diagrams.Diagram({
  width: '1500px',
  height: '600px',
  getNodeDefaults: function(node) {
    node.height = 100;
    node.width = 100;
    node.style.fill = '#6BA5D7';
    node.style.strokeColor = 'white';
    return node;
  },
  nodes: nodes,
}, '#element');

diagram.add(group);
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
```

```

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Add children To group at runtime

A childNode can be added to the specified Group at runtime by utilizing the client-side method `diagram.addChildToGroup`.

This functionality is achieved by passing the group and existing children as arguments to the method.

The following code illustrates how a child node and a group node can be passed as arguments to the method and executed at runtime.

```
`html
```

```
diagram.addChildToGroup(groupNode, childNode);
```

```
,
```

Remove children from group at runtime

A specific child from a group node can be removed at runtime by utilizing the client-side method `diagram.removeChildFromGroup`.

This functionality is achieved by passing the group and its children as arguments to the method.

The following code illustrates how a child node is removed from a group at runtime.

`html

diagram.removeChildFromGroup (groupNode, childNode);

`

INDEX.JS

```
var diagram;
var node = {
  id: 'node1', width: 150, height: 100, offsetX: 100, offsetY: 100,
  annotations: [{ content: 'Node1' }]
};
var node2 = {
  id: 'node2', width: 80, height: 130, offsetX: 200, offsetY: 200,
  annotations: [{ content: 'Node2' }]
};
var group = {
  id: 'group1', children: ['node1', 'node2']
};
var node3 = {
  id: 'node3', width: 100, height: 100, offsetX: 300, offsetY: 300,
  annotations: [{ content: 'Node3' }]
};
//Initializes diagram control
diagram = new ej.diagrams.Diagram({
  width: 900,
  height: 900,
  nodes: [node, node2, node3, group],
}, '#element');
//To Add child to specific group at Runtime
diagram.addChildToGroup(group, 'node3');
//To remove the specific children from group at runtime
diagram.removeChildFromGroup(group, 'node3');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Container

Containers are used to automatically measure and arrange the size and position of the child elements in a predefined manner. There are two types of containers available.

Canvas

- The canvas panel supports absolute positioning and provides the least layout functionality to its contained diagram elements.
- Canvas allows you to position its contained elements by using the margin and alignment properties.
- Rendering alone possible in canvas container.
- It allows elements to be either vertically or horizontally aligned.
- Child can be defined with the collection [canvas.children](#) property.
- Basic element can be defined with the collection of [basicElements](#).

The following code illustrates how to add canvas panel.

INDEX.JS

```

var diagram;
var canvas;
var child1;
var child2;
canvas = new ej.diagrams.Canvas();
canvas.pivot = {
    x: 0,
    y: 0
};
canvas.offsetX = 200;
canvas.offsetY = 100;

```

```

canvas.style.fill = '#6BA5D7';
child1 = new ej.diagrams.DiagramElement();
child1.width = 100;
child1.height = 100;
child1.margin.left = child1.margin.top = 10;
child2 = new ej.diagrams.DiagramElement();
child2.width = 100;
child2.height = 100;
child2.margin.left = 190;
child2.margin.top = 190;
canvas.children = [child1, child2];
diagram = new ej.diagrams.Diagram({
    mode: 'SVG',
    width: '1000px',
    height: '600px',
    basicElements: [canvas]
}, '#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
```

```

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Stack

- Stack panel is used to arrange its children in a single line or stack order, either vertically or horizontally.
- It controls spacing by setting margin properties of child and padding properties of group. By default, a stack panel's [orientation](#) is vertical.

The following code illustrates how to add a stack panel.

INDEX.JS

```

var diagram;
var nodes = [{
  id: "node5",
  width: 100,
  height: 100,
  offsetX: 100,
  offsetY: 100,
  style: {
    strokeColor: "#6BA5D7",
    fill: "#6BA5D7"
  },
  annotations: [{
    content: "Custom Template",
    offset: {
      y: 1
    },
    verticalAlignment: "Top"
  }]
}];
var count = 11;
var getTextElement = function(text) {
  var textElement = new ej.diagrams.TextElement();
  textElement.id = "text" + count;
  textElement.width = 50;
  textElement.height = 20;
  textElement.content = text;
  count++;
  return textElement;
};
var addRows = function(column) {
  column.children.push(getTextElement("Row1"));
  column.children.push(getTextElement("Row2"));
  column.children.push(getTextElement("Row3"));
  column.children.push(getTextElement("Row4"));
};
//Initializes diagram control
diagram = new ej.diagrams.Diagram({
  width: 900,
  height: 900,
  nodes: nodes,

```

```

setNodeTemplate: function(obj, diagram) {
    if (obj.id.indexOf("node5") !== -1) {
        var table = new ej.diagrams.StackPanel();
        table.orientation = "Horizontal";
        var column1 = new ej.diagrams.StackPanel();
        column1.children = [];
        column1.children.push(getTextElement("Column1"));
        addRows(column1);
        var column2 = new ej.diagrams.StackPanel();
        column2.children = [];
        column2.children.push(getTextElement("Column2"));
        addRows(column2);
        table.children = [column1, column2];
        return table;
    }
    return null;
},
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Diagram</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";

```

```

}
    </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Difference between a basic group and containers

| Group | Container |

| ----- | ----- |

| It arranges the child elements based on the child elements position and size properties. | Each container has a predefined behavior to measure and arrange its child elements. Canvas and stack containers are supported in the diagram. |

| The Padding, Min, and Max Size properties are not applicable for basic group. | It is applicable for container. |

| The Children's margin and alignment properties are not applicable for basic group. | It is applicable for container. |

Interaction

You can edit the group and its children at runtime. For more information about how to interact with a group, refer to [Edit Groups](#).

See Also

- [How to add annotations to the node](#)
- [How to add ports to the node](#)
- [How to enable/disable the behavior of the node](#)
- [How to add nodes to the symbol palette](#)
- [How to create diagram nodes using drawing tools](#)
- [How to perform the interaction on the group](#)

Swim lane in ##Platform_Name## Diagram control

Swimlane is a type of diagram nodes, which is typically used to visualize the relationship between a business process and the department responsible for it by focusing on the logical relationships between activities.

Create a swimlane

To create a swimlane, the type of shape should be set as [swimlane](#). By Default swimlane's are arranged vertically.

The following code example illustrates how to define a swimlane object.

INDEX.JS

```

var node = {
  shape: {
    type: 'SwimLane',
    lanes: [
      {
        id: 'stackCanvas1',
        height: 100,
      },
    ],
  },
}

```

```

        ],
        phases: [
            {
                id: 'phase1', offset: 170,
                header: { annotation: { content: 'Phase' } }
            },
        ],
        phaseSize: 20,
    },
    offsetX: 300, offsetY: 200,
    height: 200,
    width: 350
};
// initialize Diagram component
var diagram = new ej.diagrams.Diagram({
    width: '100%', height: '600px', nodes: [node]
});
diagram.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";

```

```

}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Headers

Header was the primary element for swimlanes. The [header](#) property of swimlane allows you to define its textual description and to customize its appearance.

Note: By using this header, the swimlane interaction will be performed, like selection, dragging, etc.

The following code example illustrates how to define a swimlane header.

INDEX.JS

```

var node = {
  shape: {
    type: 'SwimLane',
    orientation: 'Horizontal',
    //Intialize header to swimlane
    header: {
      annotation: { content: 'ONLINE PURCHASE STATUS', style:
{ fill: '#111111' } },
      height: 50, style: { fontSize: 11 },
    },
    lanes: [
      {
        id: 'stackCanvas1',
        height: 100,
      },
    ],
    phases: [
      {
        id: 'phase1', offset: 170,
        header: { annotation: { content: 'Phase' } }
      },
    ],
    phaseSize: 20,
  },
  offsetX: 300, offsetY: 200,
  height: 200,
  width: 350
};
// initialize Diagram component
var diagram = new ej.diagrams.Diagram({
  width: '100%', height: '600px', nodes: [node]
});
diagram.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">

```



```

<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customization of headers

The height and width of swimlane header can be customized with [weight](#) and [height](#) properties of swimlane header. set fill color of header by using the [style](#) property. The orientation of swimlane can be customized with the [orientation](#) property of the header.

Note: By default the swimlane orientation has Horizontal.

The following code example illustrates how to customize the swimlane header.

INDEX.JS

```

var node = {
    shape: {
        type: 'SwimLane',
        orientation: 'Horizontal',
        // customize the swimlane header
        header: {
            annotation: { content: 'SALES PROCESS FLOW CHART', },
            height: 70, style: { fontSize: 11 }, style: { fill: 'pink' }
        },
    },
},

```

```

        lanes: [
            {
                id: 'stackCanvas1',
                height: 100,
            },
        ],
        phases: [
            {
                id: 'phase1', offset: 170,
                header: { annotation: { content: 'Phase' } }
            },
        ],
        phaseSize: 20,
    },
    offsetX: 300, offsetY: 200,
    height: 200,
    width: 350
};
// initialize Diagram component
var diagram = new ej.diagrams.Diagram({
    width: '100%', height: '600px', nodes: [node]
});
diagram.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>

```

```

    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Dynamic customization of swimlane header

You can customize the swimlane header style and text properties dynamically. The following code illustrates how to dynamically customize the lane header.

INDEX.JS

```

var node = {
    shape: {
        type: 'SwimLane',
        orientation: 'Horizontal',
        // customize the swimlane header
        header: {
            annotation: { content: 'SALES PROCESS FLOW CHART', },
            height: 70, style: { fontSize: 11 }, style: { fill: 'pink' }
        },
        lanes: [
            {
                id: 'stackCanvas1',
                height: 100,
            },
            {
                phases: [
                    {
                        id: 'phase1', offset: 170,
                        header: { annotation: { content: 'Phase' } }
                    },
                ],
                phaseSize: 20,
            },
        ],
        offsetX: 300, offsetY: 200,
        height: 200,
        width: 350
    };
// initialize Diagram component
var diagram = new ej.diagrams.Diagram({
    width: '100%', height: '600px', nodes: [node]
});
diagram.appendTo('#element');
diagram.nodes[0].shape.header.style.fill = 'red'
diagram.dataBind();

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>

```

```

<title>EJ2 Diagram</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

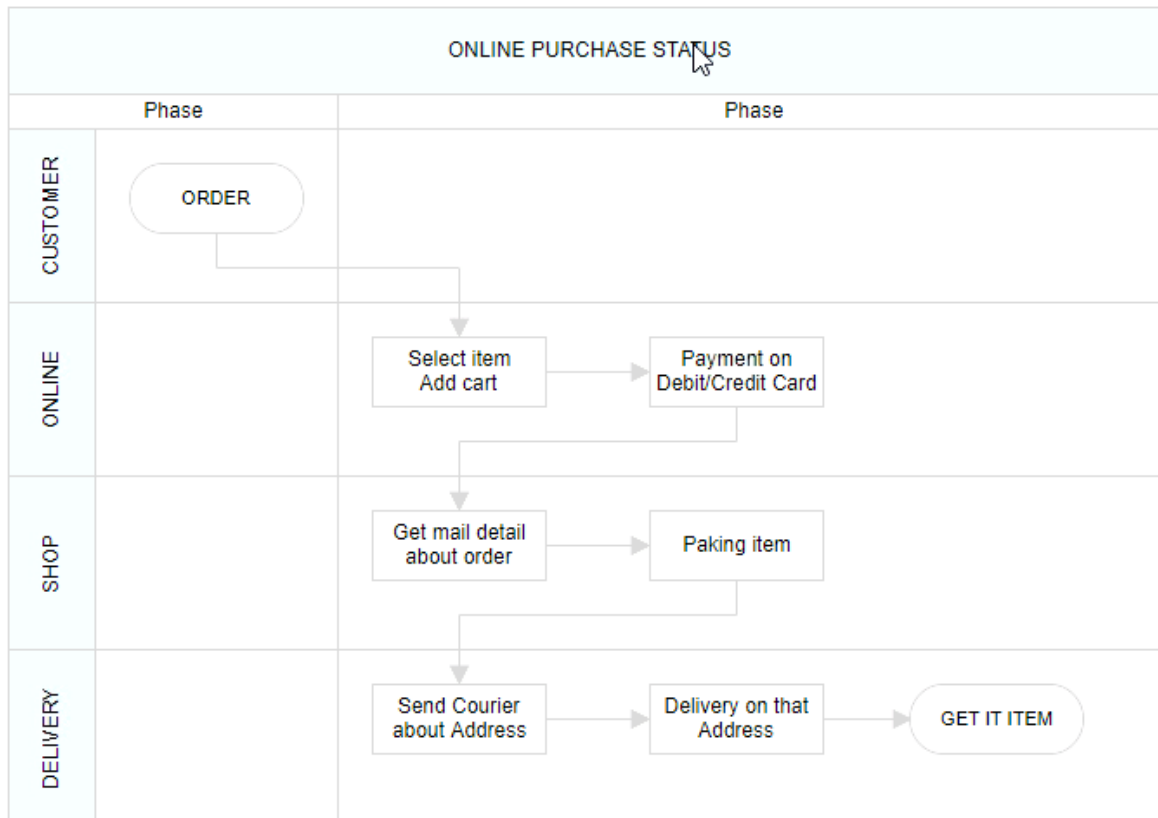
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Header editing

Diagram provides the support to edit swimlane headers at runtime. We achieve the header editing by double click event. Double clicking the header label will enables the editing of that. The following image illustrates how to edit the swimlane header.



Lanes

Lane is a functional unit or a responsible department of a business process that helps to map a process within the functional unit or in between other functional units.

The number of [lanes](#) can be added to swimlane. The lanes are automatically stacked inside swimlane based on the order they are added.

Create an empty lane

- The lane `id` is used to define the name of the lane and its further used to find the lane at runtime and do any customization.
- We can provide additional information to the lane by using the [addInfo](#) property of the lane.

The following code example illustrates how to define a swimlane with lane.

INDEX.JS

```

var node = {
  shape: {
    type: 'SwimLane',
    orientation: 'Horizontal',
    //Intialize header to swimlane
    header: {
      annotation: { content: 'ONLINE PURCHASE STATUS', style:
{ fill: '#111111' } },
      height: 50, style: { fontSize: 11 },

```

```

    },
    // initialize the lane of swimlane
    lanes: [
        {
            id: 'stackCanvas1',
            // set the lane height
            height: 100,
        },
    ],
    phases: [
        {
            id: 'phase1', offset: 170,
            header: { annotation: { content: 'Phase' } }
        },
    ],
    phaseSize: 20,
},
offsetX: 300, offsetY: 200,
height: 200,
width: 350
};
// initialize Diagram component
var diagram = new ej.diagrams.Diagram({
    width: '100%', height: '600px', nodes: [node]
});
diagram.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```

```

<div id="container">
  <div id="element"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Create lane header

- The [header](#) property of lane allows you to textually describe the lane and to customize the appearance of the description.

The following code example illustrates how to define a lane header.

INDEX.JS

```

var node = {
  shape: {
    type: 'SwimLane',
    orientation: 'Horizontal',
    //Intialize header to swimlane
    header: {
      annotation: { content: 'ONLINE PURCHASE STATUS', style:
{ fill: '#111111' } } },
      height: 50, style: { fontSize: 11 },
    },
    // Intialize lane to swimlane
    lanes: [
      {
        id: 'stackCanvas1',
        height: 100,
        // Intialize header to lane
        header: {
          annotation: { content: 'CUSTOMER' }, width: 50,
          style: { fontSize: 11 }
        },
      },
    ],
    phases: [
      {
        id: 'phase1', offset: 170,
        header: { annotation: { content: 'Phase' } } }
    ],
    phaseSize: 20,
  },
  offsetX: 300, offsetY: 200,
  height: 200,
  width: 350
}

```

```

    };
    // initialize Diagram component
    var diagram = new ej.diagrams.Diagram({
        width: '100%', height: '600px', nodes: [node]
    });
    diagram.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Diagram</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
    <script>
    var ele = document.getElementById('container');
    if(ele) {
        ele.style.visibility = "visible";
    }
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

Customizing lane header

- The size of lane can be controlled by using [width](#) and [height](#) properties of lane.
- The appearance of lane can be set by using the [style](#) properties.

The following code example illustrates how to customize the lane header.

INDEX.JS

```
var node = {
  shape: {
    type: 'SwimLane',
    orientation: 'Horizontal',
    //Intialize header to swimlane
    header: {
      annotation: { content: 'ONLINE PURCHASE STATUS', style:
{ fill: '#111111' } },
      height: 50, style: { fontSize: 11 },
    },
    lanes: [
      {
        id: 'stackCanvas1',
        height: 100,
        // customization of lane header
        header: {
          annotation: { content: 'Online Consumer' }, width:
30,
          style: { fontSize: 11 },style: { fill: 'red' }
        },
      },
    ],
    phases: [
      {
        id: 'phase1', offset: 170,
        header: { annotation: { content: 'Phase' } }
      },
    ],
    phaseSize: 20,
  },
  offsetX: 300, offsetY: 200,
  height: 200,
  width: 350
};
// initialize Diagram component
var diagram = new ej.diagrams.Diagram({
  width: '100%', height: '600px', nodes: [node]
});
diagram.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Dynamic customization of lane header

You can customize the lane header style and text properties dynamically. The following code illustrates how to dynamically customize the lane header.

INDEX.JS

```

var node = {
    shape: {
        type: 'SwimLane',
        orientation: 'Horizontal',
        //Intialize header to swimlane
        header: {
            annotation: { content: 'ONLINE PURCHASE STATUS', style:
{ fill: '#111111' } },
            height: 50, style: { fontSize: 11 },
        },
        lanes: [
            {
                id: 'stackCanvas1',
                height: 100,
                // customization of lane header
                header: {
                    annotation: { content: 'Online Consumer' }, width:
30,
                    style: { fontSize: 11 },style: { fill: 'red' }
                },
            },
        ],
    },
};

```

```

        },
        ],
        phases: [
            {
                id: 'phase1', offset: 170,
                header: { annotation: { content: 'Phase' } }
            },
        ],
        phaseSize: 20,
    },
    offsetX: 300, offsetY: 200,
    height: 200,
    width: 350
};
// initialize Diagram component
var diagram = new ej.diagrams.Diagram({
    width: '100%', height: '600px', nodes: [node]
});
diagram.appendTo('#element');
var lane = diagram.nodes[0];
lane.shape.lanes[0].header.style.fill = 'blue';
diagram.dataBind();

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>

```

```
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Add lane at runtime

You can add the a lane at runtime by using the client side API method called `addLanes`. The following code illustrates how to dynamically add lane to swimlane.

You can customize the lane header style and text properties dynamically. The following code illustrates how to dynamically customize the lane header.

INDEX.JS

```
var node = {
  shape: {
    type: 'SwimLane',
    orientation: 'Horizontal',
    //Intialize header to swimlane
    header: {
      annotation: { content: 'ONLINE PURCHASE STATUS', style:
{ fill: '#111111' } },
      height: 50, style: { fontSize: 11 },
    },
    lanes: [
      {
        id: 'stackCanvas1',
        addInfo:{name:'lane1'},
        height: 100,
        header: {
          annotation: { content: 'CUSTOMER' }, width: 50,
          style: { fontSize: 11 }
        },
        // Set the children of lane
        children: [
          {
            id: 'node1',
            annotations: [
              {
                content: 'Consumer learns \n of
product',
                style: { fontSize: 11 }
              }
            ],
            margin: { left: 60, top: 30 },
            height: 40, width: 100,
          }, {
            id: 'node2',
            shape: { type: 'Flow', shape: 'Decision' },
            annotations: [
              {
```

```

        content: 'Does \n Consumer want \nthe
product',
        style: { fontSize: 11 }
    }
    ],
    margin: { left: 200, top: 20 },
    height: 60, width: 120,
    },
    ],
    },
    ],
    phases: [
        {
            id: 'phase1', offset: 170,
            header: { annotation: { content: 'Phase' } }
        },
    ],
    phaseSize: 20,
    },
    offsetX: 300, offsetY: 200,
    height: 200,
    width: 350
    };
// initialize Diagram component
var diagram = new ej.diagrams.Diagram({
    width: '100%', height: '600px', nodes: [node]
});
diagram.appendTo('#element');
var lane = [{id:"lane1",height:100,}];
diagram.addLanes(diagram.nodes[0],lane,1);

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>

```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Add children to lane

To add nodes to lane, you should add [children](#) collection of the lane.

The following code example illustrates how to add nodes to lane.

INDEX.JS

```

var node = {
    shape: {
        type: 'SwimLane',
        orientation: 'Horizontal',
        //Intialize header to swimlane
        header: {
            annotation: { content: 'ONLINE PURCHASE STATUS', style:
{ fill: '#111111' } },
            height: 50, style: { fontSize: 11 },
        },
        lanes: [
            {
                id: 'stackCanvas1',
                addInfo:{name:'lane1'},
                height: 100,
                header: {
                    annotation: { content: 'CUSTOMER' }, width: 50,
                    style: { fontSize: 11 }
                },
                // Set the children of lane
                children: [
                    {
                        id: 'node1',
                        annotations: [
                            {
                                content: 'Consumer learns \n of
product',
                                style: { fontSize: 11 }
                            }
                        ],
                        margin: { left: 60, top: 30 },
                        height: 40, width: 100,

```

```

        }, {
            id: 'node2',
            shape: { type: 'Flow', shape: 'Decision' },
            annotations: [
                {
                    content: 'Does \n Consumer want \nthe
product',
                    style: { fontSize: 11 }
                }
            ],
            margin: { left: 200, top: 20 },
            height: 60, width: 120,
        },
    ],
    },
    ],
    phases: [
        {
            id: 'phase1', offset: 170,
            header: { annotation: { content: 'Phase' } }
        },
    ],
    phaseSize: 20,
    },
    offsetX: 300, offsetY: 200,
    height: 200,
    width: 350
    };
// initialize Diagram component
var diagram = new ej.diagrams.Diagram({
    width: '100%', height: '600px', nodes: [node]
});
diagram.appendTo('#element');
var lane = [{id:"lane1",height:100,}];
diagram.addLanes(diagram.nodes[0],lane,1);

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Diagram</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">

```

```
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

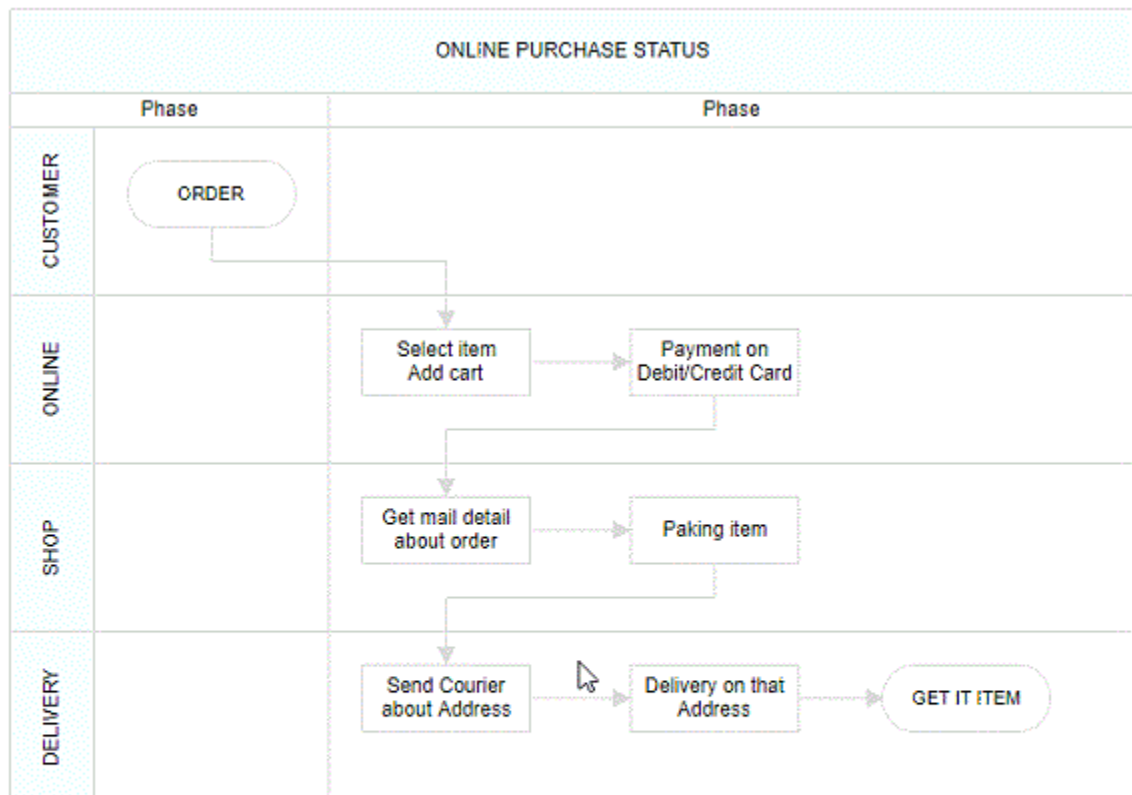
    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Lane interaction

Resizing lane

- Lane can be resized in the bottom and left direction.
- Lane can be resized by using resize selector of the lane.
- Once you can resize the lane, the swimlane will be resized automatically.
- The lane can be resized either resizing the selector or the tight bounds of the child object. If the child node move to edge of the lane it can be automatically resized. The following image

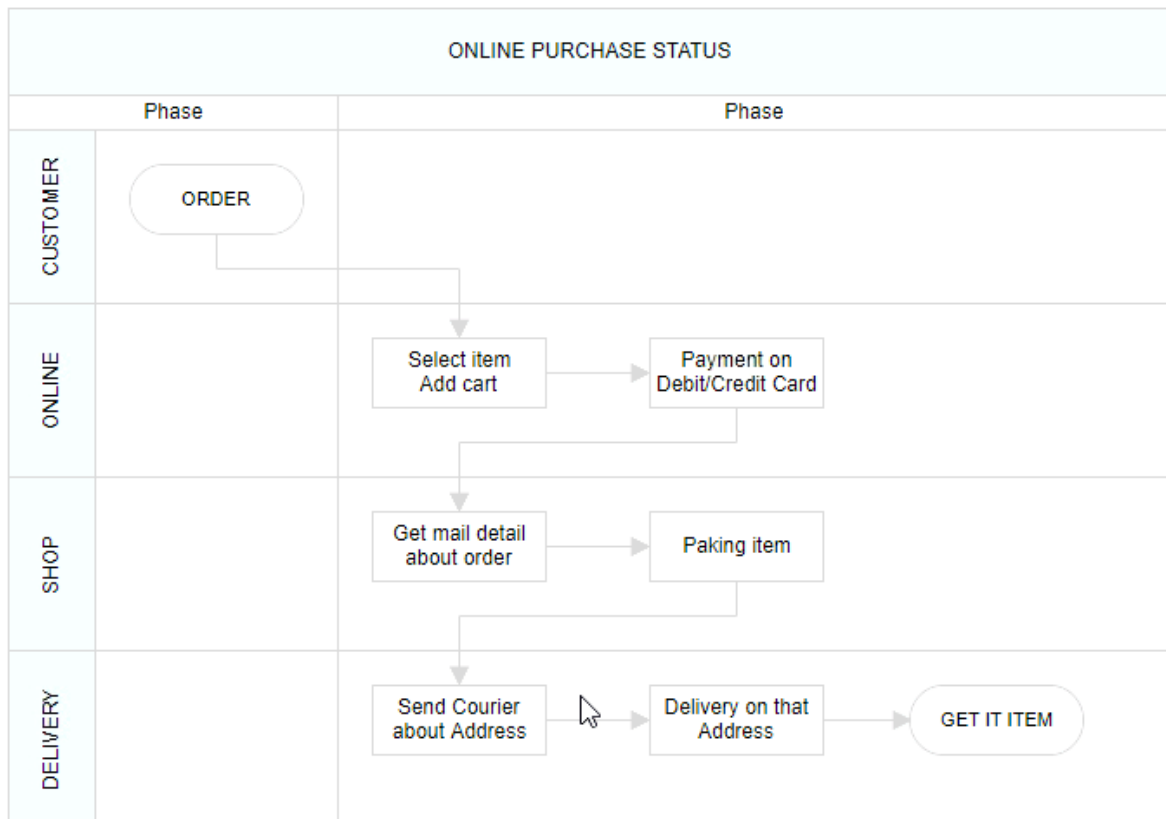
illustrates how resize the lane.



Lane swapping

- Lanes can be swapped using drag the lanes over another lane.

- Helper should intimate the insertion point while lane swapping. The following image illustrates how swapping the lane.



Disable Swimlane Lane swapping

You can disable swimlane lane swapping by using the property called `canMove`.

The following code illustrates how to disable swimlane lane swapping.

INDEX.JS

```

var node = {
  shape: {
    type: 'SwimLane',
    orientation: 'Horizontal',
    //Intialize header to swimlane
    header: {
      annotation: { content: 'ONLINE PURCHASE STATUS', style:
{ fill: '#111111' } },
      height: 50, style: { fontSize: 11 },
    },
    lanes: [
      {
        id: 'stackCanvas1',
        height: 100,
        header: {
          annotation: { content: 'CUSTOMER' }, width: 50,
          style: { fontSize: 11 }
        },
      },
    ],
  },
}

```

```

        canMove: false,
        // Set the children of lane
        children: [
            {
                id: 'node1',
                annotations: [
                    {
                        content: 'Consumer learns \n of
product',
                        style: { fontSize: 11 }
                    }
                ],
                margin: { left: 60, top: 30 },
                height: 40, width: 100,
            }, {
                id: 'node2',
                shape: { type: 'Flow', shape: 'Decision' },
                annotations: [
                    {
                        content: 'Does \n Consumer want \nthe
product',
                        style: { fontSize: 11 }
                    }
                ],
                margin: { left: 200, top: 20 },
                height: 60, width: 120,
            },
        ],
    },
    ],
    phases: [
        {
            id: 'phase1', offset: 170,
            header: { annotation: { content: 'Phase' } }
        },
    ],
    phaseSize: 20,
},
offsetX: 300, offsetY: 200,
height: 200,
width: 350
};
// initialize Diagram component
var diagram = new ej.diagrams.Diagram({
    width: '100%', height: '600px', nodes: [node]
});
diagram.appendTo('#element');
var lane = [{id:"lane1",height:100,canMove: false}];
diagram.addLanes(diagram.nodes[0],lane,1);

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">

```

```

<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

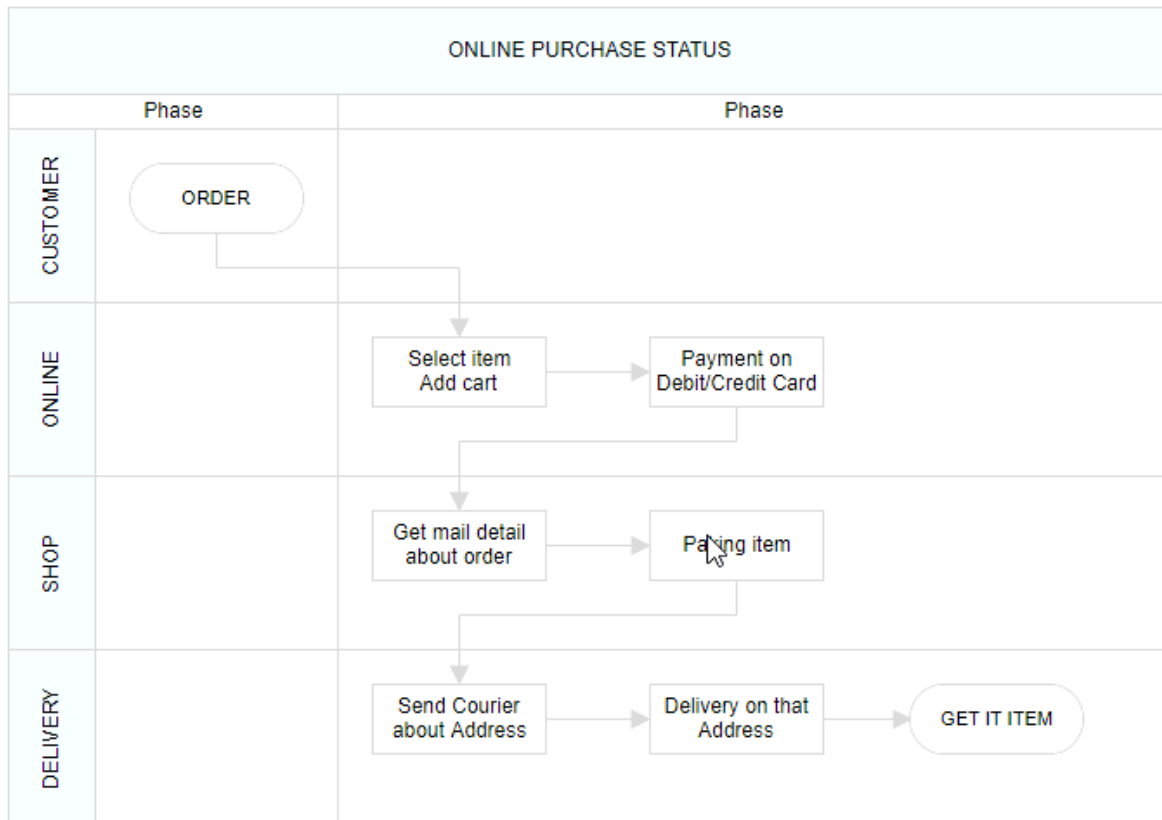
Resize helper

- The special resize helper will be used to resize the lanes.
- The resize cursor will be available on the left and bottom direction alone.
- Once resize the lane the swimlane will be resized automatically

Children interaction in lanes

- You can resize the child node within swimlanes.
- You can drag the child nodes within lane.
- Interchange the child nodes from one lane to another lane.
- Drag and drop the child nodes from lane to diagram.
- Drag and drop the child nodes from diagram to lane.
- Based on the child node interactions, the lane size should be updated.

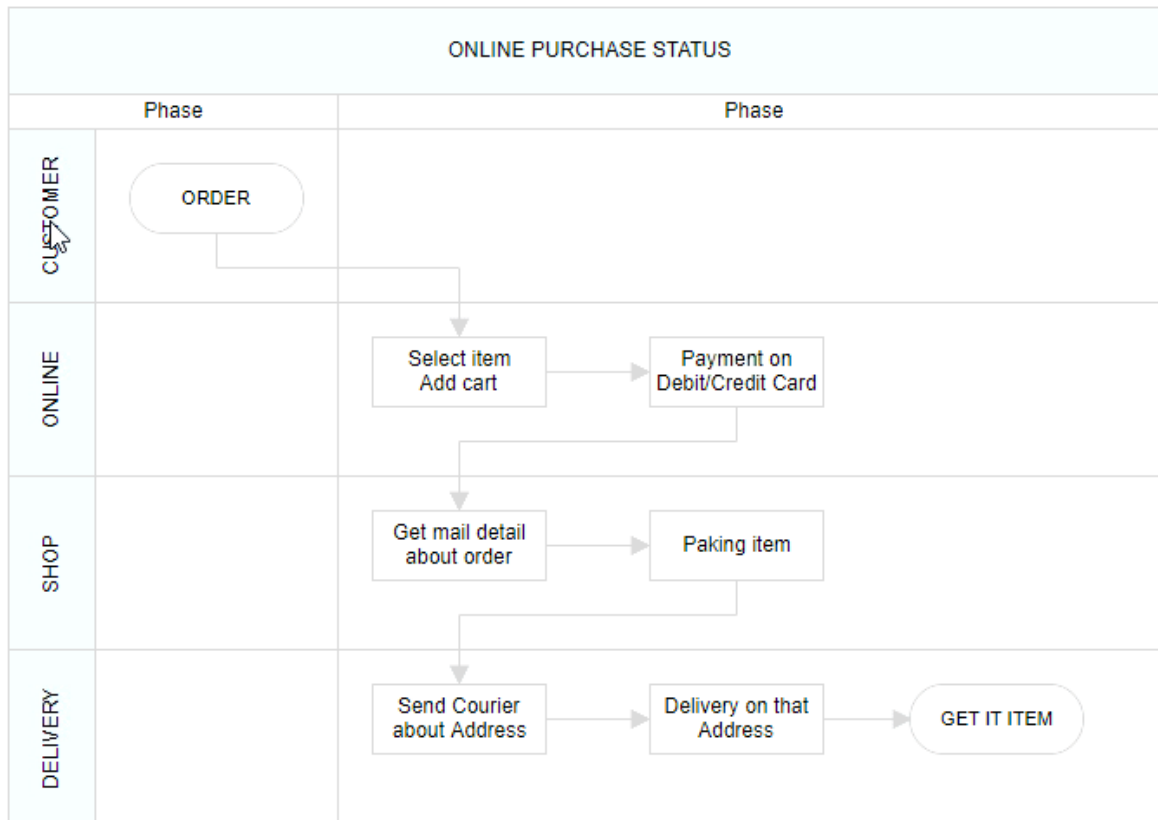
The following image illustrates children interaction in lane.



Lane header editing

Diagram provides the support to edit Lane headers at runtime. We achieve the header editing by double click event. Double clicking the header label will enables the editing of that.

The following image illustrates how to edit the lane header.



Phase

Phase are the subprocess which will split each lanes as horizontally or vertically based on the swimlane orientation. The multiple number of [Phase](#) can be added to swimlane.

The following code example illustrates how to add phase at swimlane.

INDEX.JS

```

var node = {
  shape: {
    type: 'SwimLane',
    orientation: 'Horizontal',
    //Intialize header to swimlane
    header: {
      annotation: { content: 'ONLINE PURCHASE STATUS', style:
{ fill: '#111111' } },
      height: 50, style: { fontSize: 11 },
    },
    lanes: [
      {
        id: 'stackCanvas1',
        height: 100,
        header: {
          annotation: { content: 'CUSTOMER' }, width: 50,
          style: { fontSize: 11 }
        },
        children: [

```

```

        {
            id: 'Order',
            margin: { left: 60, top: 20 },
            height: 40, width: 100
        }
    ],
    },
],
phases: [
    {
        id: 'phase1', offset: 120,
        header: { annotation: { content: 'Phase' } }
    }, {
        id: 'phase2', offset: 200,
        header: { annotation: { content: 'Phase' } }
    },
],
phaseSize: 20,
},
offsetX: 300, offsetY: 200,
height: 200,
width: 350
};
// initialize Diagram component
var diagram = new ej.diagrams.Diagram({
    width: '100%', height: '600px', nodes: [node]
});
diagram.appendTo('#element');
diagram.nodes[0].shape.header.style.fill = 'red'
diagram.dataBind();

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>

```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Dynamically add phase to Lane

You can add the a phase at runtime by using client side API method called **addPhases**. The following code example illustrates how to add phase at run time.

INDEX.JS

```

var node = {
    shape: {
        type: 'SwimLane',
        orientation: 'Horizontal',
        //Intialize header to swimlane
        header: {
            annotation: { content: 'ONLINE PURCHASE STATUS', style:
{ fill: '#111111' } },
            height: 50, style: { fontSize: 11 },
        },
        lanes: [
            {
                id: 'stackCanvas1',
                height: 100,
                header: {
                    annotation: { content: 'CUSTOMER' }, width: 50,
                    style: { fontSize: 11 }
                },
                children: [
                    {
                        id: 'Order',
                        margin: { left: 60, top: 20 },
                        height: 40, width: 100
                    }
                ]
            },
        ],
        phases: [
            {
                id: 'phase1', offset: 120,
                addInfo:{name:'phase1'},
                header: { annotation: { content: 'Phase' } }
            },
        ],
    },
}

```



```

        id: 'phase2', offset: 200,
        header: { annotation: { content: 'Phase' } }
    },
    ],
    phaseSize: 20,
    },
    offsetX: 300, offsetY: 200,
    height: 200,
    width: 350
    };
// initialize Diagram component
var diagram = new ej.diagrams.Diagram({
    width: '100%', height: '600px', nodes: [node]
});
diagram.appendTo('#element');
var phase = [{
    id: 'phase3', offset: 220,
    header: { annotation: { content: 'Phase' } }
}]
diagram.addPhases(diagram.nodes[0], phase);

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Diagram</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');

```

```

if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customizing phase

- The length of region can be set by using the [offset](#) property of the phase.
- Every phase region can be textually described with the [header](#) property of the phase.
- You can increase the width of phase by using [phaseSize](#) property of swimlane.
- We can provide additional information to the phase by using the [addInfo](#) property of the phase.

The following code example illustrates how to customize the phase in swimlane.

INDEX.JS

```

var node = {
  shape: {
    type: 'SwimLane',
    orientation: 'Horizontal',
    //Intialize header to swimlane
    header: {
      annotation: { content: 'ONLINE PURCHASE STATUS', style:
{ fill: '#111111' } },
      height: 50, style: { fontSize: 11 },
    },
    lanes: [
      {
        id: 'stackCanvas1',
        height: 100,
        header: {
          annotation: { content: 'CUSTOMER' }, width: 50,
          style: { fontSize: 11 }
        },
        children: [
          {
            id: 'Order',
            margin: { left: 60, top: 20 },
            height: 40, width: 100
          }
        ]
      },
    ],
  },
  phases: [
    {
      id: 'phase1', offset: 120,
      addInfo:{name:'phase1'},
      header: { annotation: { content: 'Phase' } }
    },{
      id: 'phase2', offset: 200,
      header: { annotation: { content: 'Phase' } }
    },
  ],
}

```

```

        phaseSize: 20,
    },
    offsetX: 300, offsetY: 200,
    height: 200,
    width: 350
};
// initialize Diagram component
var diagram = new ej.diagrams.Diagram({
    width: '100%', height: '600px', nodes: [node]
});
diagram.appendTo('#element');
var phase = [{
    id: 'phase3', offset: 220,
    header: { annotation: { content: 'Phase' } }
}]
diagram.addPhases(diagram.nodes[0], phase);

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Diagram</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>

```

```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Phase interaction

Resizing

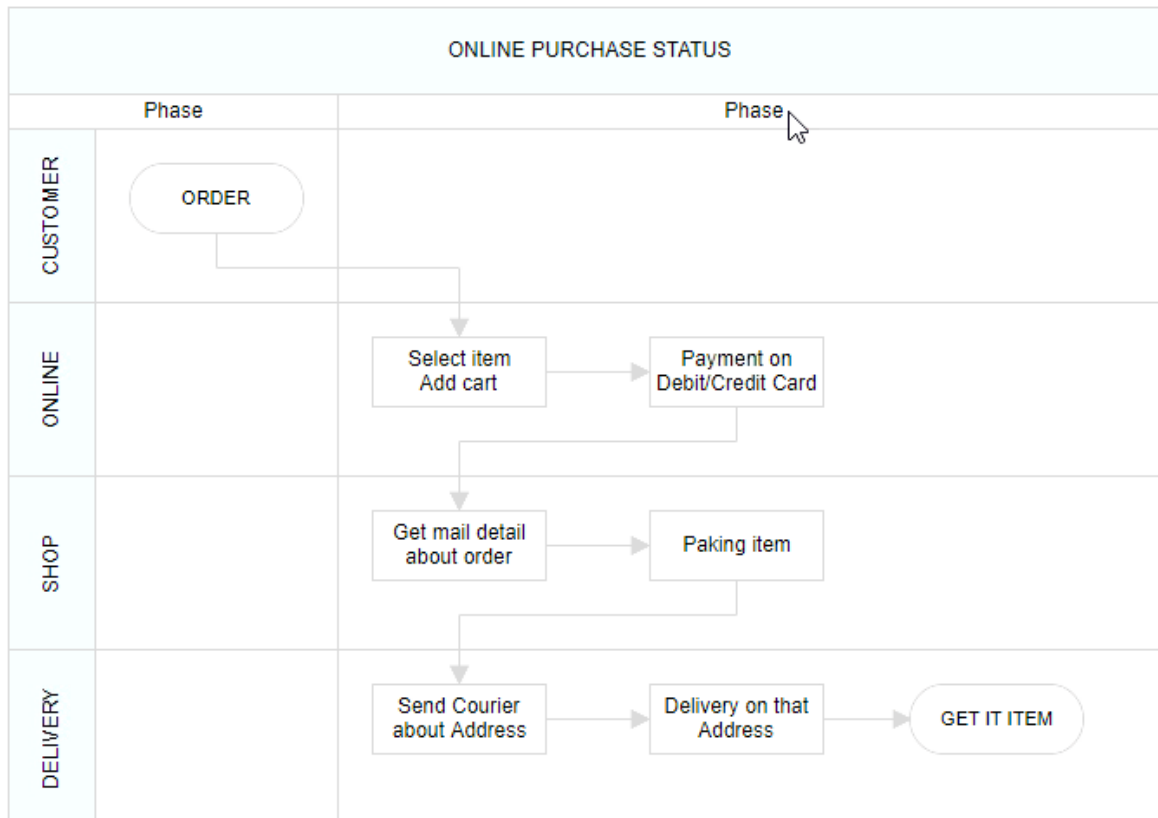
- The phase can be resized by using its selector.
- You must select the phase header to enable the phase selection.
- Once the phase can be resized, the lane size will be updated automatically.

Resizing helper

- The special resize selector will be used to resize the phase.
- The resize cursor will be available on the left and bottom direction for horizontal, and the top and bottom direction for vertical swimlane.

Phase header editing

Diagram provides the support to edit phase headers at runtime. We achieve the header editing by double click event. Double clicking the header label will enables the editing of that. The following image illustrates how to edit the swimlane header. The following image illustrates how to edit the phase header.



Add swimlane to palette

Diagram provides the support to add swimlane and phases to symbol palette. The following code sample illustrate how to add swimlane and phases to palette.

INDEX.JS

```
/**
 * Default symbol palette sample
 */
//Initialize the flowshapes for the symbol palatte
var swimlaneShapes = [
    {
        id: 'stackCanvas1',
        shape: {
            type: 'SwimLane', lanes: [
                {
                    id: 'lane1',
                    style: { strokeColor: 'black' }, height: 60,
width: 150,
                    header: { width: 50, height: 50, style: {
strokeColor: 'black', fontSize: 11 } },
                }
            ],
            orientation: 'Horizontal', isLane: true
        },
        height: 60,
        width: 140,
        offsetX: 70,
        offsetY: 30,
    }, {
        id: 'stackCanvas2',
        shape: {
            type: 'SwimLane',
            lanes: [
                {
                    id: 'lane1',
                    style: { strokeColor: 'black' }, height: 150,
width: 60,
                    header: { width: 50, height: 50, style: {
strokeColor: 'black', fontSize: 11 } },
                }
            ],
            orientation: 'Vertical', isLane: true
        },
        height: 140,
        width: 60,
        // style: { fill: '#f5f5f5' },
        offsetX: 70,
        offsetY: 30,
    }, {
        id: 'verticalPhase',
        shape: {
            type: 'SwimLane',
            phases: [{ style: { strokeWidth: 1, strokeDashArray:
'3,3', strokeColor: '#A9A9A9' } }, {}],
            annotations: [{ text: '' }],
            orientation: 'Vertical', isPhase: true
        }
    }
];
```

```

        },
        height: 60,
        width: 140
    }, {
        id: 'horizontalPhase',
        shape: {
            type: 'SwimLane',
            phases: [{ style: { strokeWidth: 1, strokeDashArray:
'3,3', strokeColor: '#A9A9A9' }, }],
            annotations: [{ text: '' }],
            orientation: 'Horizontal', isPhase: true
        },
        height: 60,
        width: 140
    }
    ];
function setPaletteNodeDefaults(node) {
    node.width = 100;
    node.height = 100;
    node.style.strokeColor = '#3A3A3A';
}
var palette = new ej.diagrams.SymbolPalette({
    expandMode: 'Multiple',
    palettes: [{
        id: 'swimlane',
        expanded: true,
        symbols: swimlaneShapes,
        title: 'Swimlane Shapes'
    }, ],
    symbolPreview: {
        height: 100,
        width: 100,
        offset: {
            x: 0.5,
            y: 0.5
        }
    },
    symbolMargin: {
        left: 12,
        right: 12,
        top: 12,
        bottom: 12
    },
    enableSearch: true,
    getNodeDefaults: setPaletteNodeDefaults,
    getSymbolInfo: (symbol) => {
        return {
            fit: true
        };
    }
});
palette.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
```

```

<title>EJ2 Diagram</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

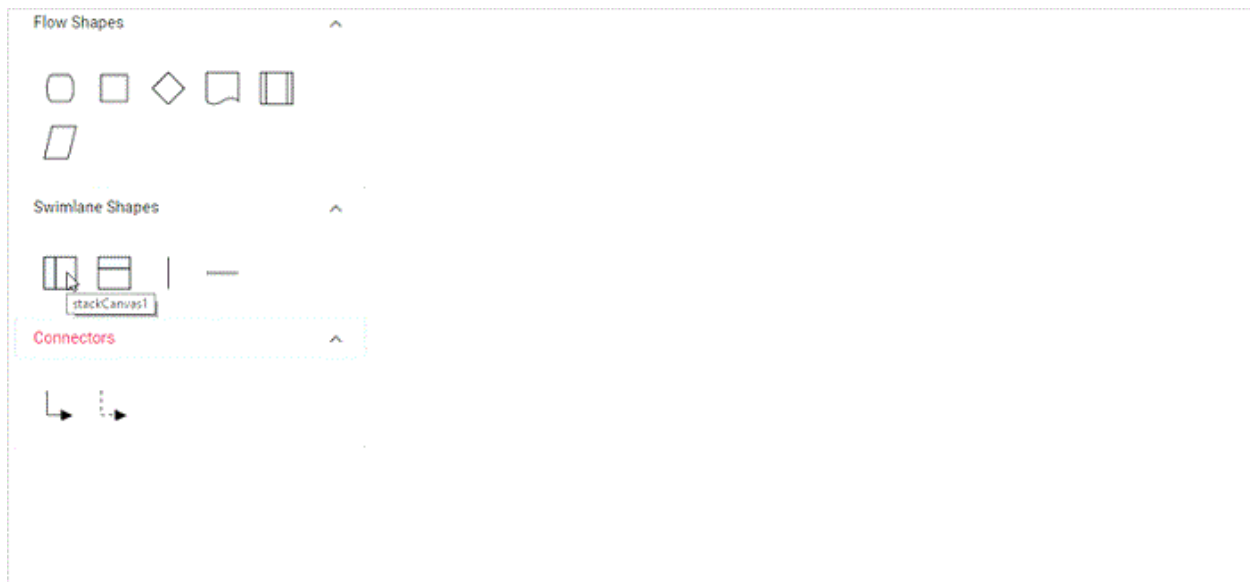
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Drag and drop swimlane to palette

- The drag and drop support for swimlane shapes has been provided.
- When you drag and drop the lane shape, if the diagram already contains swimlane with the same orientation, the lane will be added and stacked inside a swimlane based on the order. Otherwise, it will be added a new swimlane.
- The phase will only drop on swimlane shape with same orientation. The following image illustrates how to drag symbol from palette.

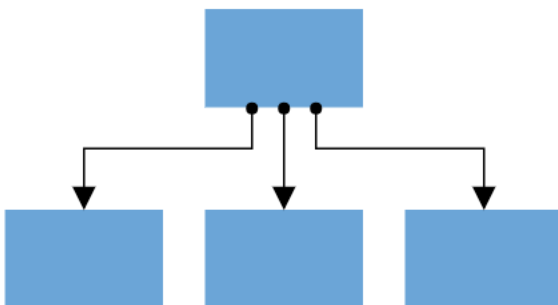


Limitations

- Connectors cannot be canceled when added directly to swimlane. we must initialize the connector through connector collection.
- We cannot edit the phase line style.

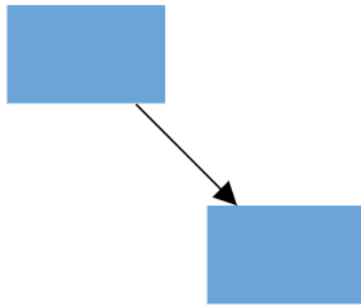
Ports in ##Platform_Name## Diagram control

Diagram provides support to define custom ports for making connections.

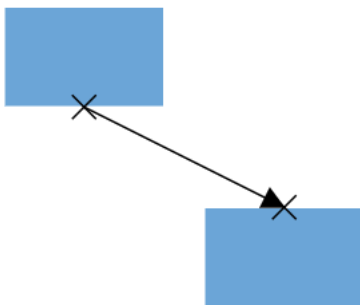


<!-- markdownlint-disable MD033 -->

When a connector is connected between two nodes, its end points are automatically docked to the node's nearest boundary as shown in the following image.



Ports act as the connection points of the node and allows to create connections with only those specific points as shown in the following image.



Create port

Add ports when initializing nodes

To add a connection port, define the port object and add it to node's ports collection. The `offset` property of port accepts an object of fractions and used to determine the position of ports. The following code illustrates how to add ports when initializing the node.

INDEX.TS

```
import {
  Diagram,
  NodeModel,
  PortVisibility
} from '@syncfusion/ej2-diagrams';
// A node is created and stored in nodes array.
let node: NodeModel = {
  // Position of the node
  offsetX: 250,
  offsetY: 250,
  // Size of the node
  width: 100,
  height: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white'
  },
  // Initialize port collection
  ports: [{
```

```

        // Sets the position for the port
        offset: {
            x: 0.5,
            y: 0.5
        },
        visibility: PortVisibility.Visible
    }]
};
// initialize diagram component
let diagram: Diagram = new Diagram({
    width: '100%',
    height: '600px',
    // Add node
    nodes: [node]
});
// render initialized diagram
diagram.appendTo('#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Diagram</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
}
```

```

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Add ports at runtime

Add ports at runtime by using the client-side method [addPorts](#). The following code illustrates how to add ports to node at runtime.

The port's ID property is used to define the unique ID for the port and its further used to find the port at runtime.

If ID is not set, then default ID is automatically set.

INDEX.TS

```

import { Diagram, NodeModel, PointPortModel, PortVisibility } from
 '@syncfusion/ej2-diagrams';
// A node is created and stored in nodes array.
let node: NodeModel = {
  // Position of the node
  offsetX: 250,
  offsetY: 250,
  // Size of the node
  width: 100,
  height: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white'
  },
};
// Initialize port collection
let port: PointPortModel[] = [{
  id: 'port1',
  offset: {
    x: 0,
    y: 0.5
  },
  visibility: PortVisibility.Visible
}, {
  id: 'port2',
  offset: {
    x: 1,
    y: 0.5
  },
  visibility: PortVisibility.Visible
},
{
  id: 'port3',
  offset: {
    x: 0.5,
    y: 0
  },
  visibility: PortVisibility.Visible
},
{
  id: 'port4',

```

```

        offset: {
            x: 0.5,
            y: 1
        },
        visibility: PortVisibility.Visible
    }
};
// initialize diagram component
let diagram: Diagram = new Diagram({
    width: '100%',
    height: '600px',
    nodes: [node]
});
// render initialized diagram
diagram.appendTo('#element');
// Method to add ports through run time
diagram.addPorts(diagram.nodes[0], port);

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Diagram</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}

```

```
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Remove ports at runtime

Remove ports at runtime by using client-side method [removePorts](#). Refer to the following example which shows how to remove ports at runtime.

INDEX.TS

```
import { Diagram, NodeModel, PointPortModel, PortVisibility } from
 '@syncfusion/ej2-diagrams';
// A node is created and stored in nodes array.
let node: NodeModel = {
  // Position of the node
  offsetX: 250,
  offsetY: 250,
  // Size of the node
  width: 100,
  height: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white'
  },
  // Initialize port collection
  ports: [{
    id: 'port1',
    offset: {
      x: 0,
      y: 0.5
    },
    visibility: PortVisibility.Visible
  },
  {
    id: 'port2',
    offset: {
      x: 1,
      y: 0.5
    },
    visibility: PortVisibility.Visible
  },
  {
    id: 'port3',
    offset: {
      x: 0.5,
      y: 0
    },
    visibility: PortVisibility.Visible
  },
  {
    id: 'port4',
    offset: {
      x: 0.5,
      y: 1
    },
    visibility: PortVisibility.Visible
  }
  ]
}
```

```

    }
  ]
};
// initialize diagram component
let diagram: Diagram = new Diagram({
  width: '100%',
  height: '600px',
  nodes: [node]
});
// render initialized diagram
diagram.appendTo('#element');
let ports: PointPortModel[] = [{
  id: 'port1',
}, {
  id: 'port2',
}, {
  id: 'port3',
}, {
  id: 'port4',
}]
diagram.removePorts(diagram.nodes[0], ports);

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
</script>

```

```

var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Update port at runtime

You can change any port properties at runtime and update it through the client-side method [dataBind](#).

The following code example illustrates how to change the port properties.

INDEX.TS

```

import { Diagram, NodeModel, PortVisibility } from '@syncfusion/ej2-
diagrams';
// A node is created and stored in nodes array.
let node: NodeModel = {
    // Position of the node
    offsetX: 250,
    offsetY: 250,
    // Size of the node
    width: 100,
    height: 100,
    style: {
        fill: '#6BA5D7',
        strokeColor: 'white'
    },
    // Initialize port collection
    ports: [{
        offset: {
            x: 0.5,
            y: 0.5
        },
        visibility: PortVisibility.Visible
    }]
};
// initialize Diagram component
let diagram: Diagram = new Diagram({
    width: '100%',
    height: '600px',
    // Add node
    nodes: [node]
});
// render initialized Diagram
diagram.appendTo('#element');
diagram.nodes[0].ports[0].offset = {
    x: 1,
    y: 1
};
diagram.dataBind();

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>

```

```

<title>EJ2 Diagram</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Appearance

- The shape of port can be changed by using its shape property. To explore the different types of port shapes, refer to Port Shapes. If you need to render a custom shape, then you can set shape as path and define path using path data property of port.
- The appearance of ports can be customized by using [strokeColor](#), [strokeWidth](#), and [fill](#) properties of the port.
- Customize the port size by using the [width](#) and [height](#) properties of port.
- The ports [visibility](#) property allows you to define, when the port should be visible.

The following code illustrates how to change the appearance of port.

INDEX.TS


```

import { Diagram, NodeModel, PortVisibility } from '@syncfusion/ej2-
diagrams';
// A node is created and stored in nodes array.
let node: NodeModel = {
    // Position of the node
    offsetX: 250,
    offsetY: 250,
    // Size of the node
    width: 100,
    height: 100,
    style: {
        fill: '#6BA5D7',
        strokeColor: 'white'
    },
    // Initialize port collection
    ports: [{
        offset: {
            x: 1,
            y: 0.5
        },
        visibility: PortVisibility.Visible,
        //Set the style for the port
        style: {
            fill: 'red',
            strokeWidth: 2,
            strokeColor: 'black'
        },
        width: 12,
        height: 12,
        // Sets the shape of the port as Circle
        shape: 'Circle'
    }]
};
// initialize Diagram component
let diagram: Diagram = new Diagram({
    width: '100%',
    height: '600px',
    // Add node
    nodes: [node]
});
// render initialized Diagram
diagram.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Offset

The offset property of port is used to align the port based on fractions. 0 represents top/left corner, 1 represents bottom/right corner, and 0.5 represents half of width/height.

Constraints

The constraints property allows to enable/disable certain behaviors of ports. For more information about port constraints, refer to [Port Constraints](#).

Grid lines in ##Platform_Name## Diagram control

Gridlines are the pattern of lines drawn behind the diagram elements. It provides a visual guidance while dragging or arranging the objects on the diagram surface.

The model's [snapSettings](#) property is used to customize the gridlines and control the snapping behavior in the diagram.

Customize the gridlines visibility

The [snapSettings.snapConstraints](#) enables you to show/hide the gridlines. The following code example illustrates how to show or hide gridlines.

If you need to enable snapping, then inject snapping module into the diagram.

INDEX.JS

```

var snapSettings = {
    constraints: (ej.diagrams.SnapConstraints.ShowLines)
};

```

```
var diagram = new ej.diagrams.Diagram({
  width: '800px',
  height: '500px',
  snapSettings: snapSettings
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

To show only horizontal/vertical gridlines or to hide gridlines, refer to [Constraints](#).

Appearance

The appearance of the gridlines can be customized by using a set of predefined properties.

- The [horizontalGridLines](#) and the [verticalGridLines](#) properties allow to customize the appearance of the horizontal and vertical gridlines respectively.

- The horizontal gridlines [lineColor](#) and [lineDashArray](#) properties are used to customizes the line color and line style of the horizontal gridlines.
- The vertical gridlines [lineColor](#) and [lineDashArray](#) properties are used to customizes the line color and line style of the vertical gridlines.

The following code example illustrates how to customize the appearance of gridlines.

INDEX.JS

```
var snapSettings = {
  constraints: (ej.diagrams.SnapConstraints.ShowLines),
  horizontalGridlines: {
    lineColor: 'blue',
    lineDashArray: '2 2'
  },
  verticalGridlines: {
    lineColor: 'blue',
    lineDashArray: '2 2'
  }
};
var diagram = new ej.diagrams.Diagram({
  width: '800px',
  height: '500px',
  snapSettings: snapSettings
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
```

```

<div id="container">
  <div id="element"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Line intervals

Thickness and the space between gridlines can be customized by using horizontal gridlines's [linesInterval](#) and vertical gridlines's [linesInterval](#) properties. In the lines interval collections, values at the odd places are referred as the thickness of lines and values at the even places are referred as the space between gridlines.

The following code example illustrates how to customize the thickness of lines and the line intervals.

INDEX.JS

```

var snapSettings = {
  constraints: (ej.diagrams.SnapConstraints.ShowLines),
  horizontalGridlines: { lineIntervals:[1.25, 14, 0.25, 15, 0.25, 15,
0.25, 15, 0.25, 15],lineColor: 'blue', lineDashArray: '2 2' },
  verticalGridlines: { lineIntervals:[1.25, 14, 0.25, 15, 0.25, 15, 0.25,
15, 0.25, 15],lineColor: 'blue', lineDashArray: '2 2' }
};
var diagram = new ej.diagrams.Diagram({
  width: '800px', height: '500px',
  snapSettings: snapSettings
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Snapping

Snap to lines

This feature allows the diagram objects to snap to the nearest intersection of gridlines while being dragged or resized. This feature enables easier alignment during layout or design.

Snapping to gridlines can be enabled/disabled with the [snapSettings.snapConstraints](#). The following code example illustrates how to enable/disable the snapping to gridlines.

INDEX.JS

```

let nodes = [{
    id: 'node1',
    width: 100,
    height: 100,
    offsetX: 100,
    offsetY: 100,
    style: {
        strokeColor: '#6BA5D7',
        fill: '#6BA5D7'
    }
}];
var snapSettings = {
    constraints: (ej.diagrams.SnapConstraints.SnapToLines) |
(ej.diagrams.SnapConstraints.ShowLines)
};
var diagram = new ej.diagrams.Diagram({
    width: '800px',
    height: '500px',
    nodes: nodes,
    getNodeDefaults: function(node) {
        node.height = 100;
        node.width = 100;
        node.style.fill = '#6BA5D7';
        node.style.strokeColor = 'white';
    }
});

```

```

        return node;
    },
    snapSettings: snapSettings
}, '#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Customization of snap intervals

By default, the objects are snapped towards the nearest gridline. The gridline or position towards where the diagram object snaps can be customized with the horizontal gridlines's [snapInterval](#) and the vertical gridlines's [snapInterval](#) properties.

INDEX.JS

```
let nodes = [{
```

```

    id: 'node1',
    width: 100,
    height: 100,
    offsetX: 100,
    offsetY: 100,
    style: {
        strokeColor: '#6BA5D7',
        fill: '#6BA5D7'
    }
  }];
var snapSettings = {
  horizontalGridlines: {
    snapIntervals: [10]
  },
  verticalGridlines: {
    snapIntervals: [10]
  },
  constraints: (ej.diagrams.SnapConstraints.All)
};
var diagram = new ej.diagrams.Diagram({
  width: '800px',
  height: '500px',
  getNodeDefaults: function(node) {
    node.height = 100;
    node.width = 100;
    node.style.fill = '#6BA5D7';
    node.style.strokeColor = 'white';
    return node;
  },
  nodes: nodes,
  snapSettings: snapSettings
}, '#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">
```



```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Snap to objects

The snap to object provides visual cues to assist with aligning and spacing diagram elements. A node can be snapped with its neighboring objects based on certain alignments. Such alignments are visually represented as smart guides.

The [snapObjectDistance](#) property allows you to define minimum distance between the selected object and the nearest object.

The [snapAngle](#) property allows you to define the snap angle by which the object needs to be rotated.

The [snapConstraints](#) property allows you to enable or disable the certain features of the snapping, refer to [snapConstraints](#).

The [snapLineColor](#) property allows you to define the color of the snapline.

INDEX.JS

```

let nodes = [{
    id: 'node1',
    width: 100,
    height: 100,
    offsetX: 100,
    offsetY: 100,
    style: {
        strokeColor: '#6BA5D7',
        fill: '#6BA5D7'
    }
}];
var snapSettings = {
    snapObjectDistance: 10,
    snapAngle: 10,
    snapLineColor: 'red'
};
var diagram = new ej.diagrams.Diagram({
    width: '800px',
    height: '500px',
    nodes: nodes,

```

```

getNodeDefaults: function(node) {
    node.height = 100;
    node.width = 100;
    node.style.fill = '#6BA5D7';
    node.style.strokeColor = 'white';
    return node;
},
snapSettings: snapSettings
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Page settings in ##Platform_Name## Diagram control

Page settings enable to customize the appearance, width, and height of the diagram page.

Page size and appearance

- The size and appearance of the diagram pages can be customized with the page settings property.
- The [width](#) and [height](#) properties of page settings define the size of the page and based on the size, the [orientation](#) will be set for the page. In addition to that, the appearance of the page can be customized with [source](#) and set of appearance specific properties.
- The [color](#) property is used to customize the background color and border color of the page.
- The [margin](#) property is used to define the page margin.
- To explore those properties, refer to [Page Settings](#).

INDEX.JS

```
var diagram;
var connector = {
  id: 'connector1',
  sourceID: 'node1',
  targetID: 'node2',
  style: {
    strokeColor: '#6BA5D7',
    fill: '#6BA5D7'
  },
};
var node = {
  id: 'node1',
  width: 100,
  style: {
    strokeColor: '#6BA5D7',
    fill: '#6BA5D7'
  },
  height: 100,
  offsetX: 100,
  offsetY: 100,
  annotations: [{
    content: 'Node1'
  }]
};
var node2 = {
  id: 'node2',
  width: 100,
  style: {
    strokeColor: '#6BA5D7',
    fill: '#6BA5D7'
  },
  height: 100,
  offsetX: 300,
  offsetY: 350,
  annotations: [{
    content: 'Node3'
  }]
};
diagram = new ej.diagrams.Diagram({
  width: '1000px',
  height: '500px',
  nodes: [node, node2],
```

```

connectors: [connector],
getNodeDefaults: function(node) {
    node.height = 100;
    node.width = 100;
    node.style.fill = '#6BA5D7';
    node.style.strokeColor = 'white';
    return node;
},
getConnectorDefaults: function(obj) {
    obj.style.strokeColor = '#6BA5D7';
    obj.style.fill = '#6BA5D7';
    obj.style.strokeWidth = 2;
    obj.targetDecorator.style.fill = '#6BA5D7';
    obj.targetDecorator.style.strokeColor = '#6BA5D7';
    return obj;
},
pageSettings: {
    orientation: 'Landscape',
    showPageBreaks: true,
    background: {
        color: 'grey'
    },
    width: 300,
    height: 300,
    margin: {
        left: 10,
        top: 10,
        bottom: 10
    },
},
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Set background image

Stretch and align the background image anywhere over the diagram area. The [source](#) property of [background](#) allows you to set the path of the image. The [scale](#) and the [align](#) properties help to stretch/align the background images.

The following code illustrates how to stretch and align the background image.

INDEX.JS

```

var diagram;
var connector = {
    id: 'connector1',
    sourceID: 'node1',
    targetID: 'node2'
};
var node = {
    id: 'node1',
    width: 100,
    height: 100,
    style: {
        strokeColor: '#6BA5D7',
        fill: '#6BA5D7'
    },
    offsetX: 100,
    offsetY: 100,
    annotations: [{
        content: 'Node1'
    }]
};
var node2 = {
    id: 'node2',
    width: 100,
    height: 100,
    style: {
        strokeColor: '#6BA5D7',
        fill: '#6BA5D7'
    },
};

```

```

        offsetX: 300,
        offsetY: 350,
        annotations: [{
            content: 'Node3'
        }]
    };
    diagram = new ej.diagrams.Diagram({
        width: '1000px',
        height: '500px',
        nodes: [node, node2],
        connectors: [connector],
        getNodeDefaults: function(node) {
            node.height = 100;
            node.width = 100;
            node.style.fill = '#6BA5D7';
            node.style.strokeColor = 'white';
            return node;
        },
        getConnectorDefaults: function(obj) {
            obj.style.strokeColor = '#6BA5D7';
            obj.style.fill = '#6BA5D7';
            obj.style.strokeWidth = 2;
            obj.targetDecorator.style.fill = '#6BA5D7';
            obj.targetDecorator.style.strokeColor = '#6BA5D7';
            return obj;
        },
        pageSettings: {
            orientation: 'Landscape',
            showPageBreaks: true,
            background: {
                source: 'https://www.w3schools.com/images/w3schools_green.jpg'
            },
            width: 300,
            height: 300,
            margin: {
                left: 10,
                top: 10,
                bottom: 10
            }
        },
    }, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Multiple page and page breaks

When multiple page is enabled, the size of the page dynamically increases or decreases in multiples of page width and height and completely fits diagram within the page boundaries. Page breaks is used as a visual guide to see how pages are split into multiple pages.

The [multiplePage](#) and [showPageBreak](#) properties of page settings allow you to enable/disable multiple pages and page breaks respectively.

The following code illustrates how to enable multiple page and page break lines.

INDEX.JS

```

var diagram;
var connector = {
    id: 'connector1',
    sourceID: 'node1',
    targetID: 'node2',
    style: {
        strokeColor: '#6BA5D7',
        fill: '#6BA5D7'
    },
};
var node = {
    id: 'node1',
    width: 100,
    height: 100,
    style: {
        strokeColor: '#6BA5D7',

```

```

        fill: '#6BA5D7'
    },
    offsetX: 100,
    offsetY: 100,
    annotations: [{
        content: 'Node1'
    }]
};
var node2 = {
    id: 'node2',
    width: 100,
    height: 100,
    style: {
        strokeColor: '#6BA5D7',
        fill: '#6BA5D7'
    },
    offsetX: 300,
    offsetY: 350,
    annotations: [{
        content: 'Node3'
    }]
};
diagram = new ej.diagrams.Diagram({
    width: '1000px',
    height: '500px',
    getNodeDefaults: function(node) {
        node.height = 100;
        node.width = 100;
        node.style.fill = '#6BA5D7';
        node.style.strokeColor = 'white';
        return node;
    },
    getConnectorDefaults: function(obj) {
        obj.style.strokeColor = '#6BA5D7';
        obj.style.fill = '#6BA5D7';
        obj.style.strokeWidth = 2;
        obj.targetDecorator.style.fill = '#6BA5D7';
        obj.targetDecorator.style.strokeColor = '#6BA5D7';
        return obj;
    },
    nodes: [node, node2],
    connectors: [connector],
    pageSettings: {
        orientation: 'Landscape',
        multiplePage: true,
        showPageBreaks: true,
        width: 300,
        height: 300,
        margin: {
            left: 10,
            top: 10,
            bottom: 10
        }
    },
}, '#element');
```


INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Boundary constraints

The diagram provides support to restrict/customize the interactive region, out of which the elements cannot be dragged, resized, or rotated. The [boundaryConstraints](#) property of page settings allows you to customize the interactive region. To explore the boundary constraints, refer to [Boundary Constraints](#).

The following code example illustrates how to define boundary constraints with respect to the page.

INDEX.JS

```

var diagram;
var connector = {
  id: 'connector1',
  sourcePoint: {
    x: 300,

```

```
        y: 100
      },
      targetPoint: {
        x: 400,
        y: 100
      }
    }
  };
  var node = {
    id: 'node1',
    width: 150,
    height: 100,
    offsetX: 100,
    offsetY: 100,
    style: {
      strokeColor: '#6BA5D7',
      fill: '#6BA5D7'
    },
  },
};
  var node2 = {
    id: 'node2',
    width: 80,
    height: 130,
    offsetX: 200,
    offsetY: 200,
    style: {
      strokeColor: '#6BA5D7',
      fill: '#6BA5D7'
    },
  },
};
  var node3 = {
    id: 'node3',
    width: 100,
    height: 75,
    offsetX: 300,
    offsetY: 350,
    style: {
      strokeColor: '#6BA5D7',
      fill: '#6BA5D7'
    },
  },
};
  diagram = new ej.diagrams.Diagram({
    width: 800,
    height: 800,
    nodes: [node, node2, node3],
    connectors: [connector],
    pageSettings: {
      boundaryConstraints: 'Page',
      width: 400,
      height: 400,
      getNodeDefaults: function(node) {
        node.height = 100;
        node.width = 100;
        node.style.fill = '#6BA5D7';
        node.style.strokeColor = 'white';
        return node;
      },
      getConnectorDefaults: function(obj) {
```

```

        obj.style.strokeColor = '#6BA5D7';
        obj.style.fill = '#6BA5D7';
        obj.style.strokeWidth = 2;
        obj.targetDecorator.style.fill = '#6BA5D7';
        obj.targetDecorator.style.strokeColor = '#6BA5D7';
        return obj;
    },
    showPageBreaks: true,
},
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Scroll settings in ##Platform_Name## Diagram control

The diagram can be scrolled by using the vertical and horizontal scrollbars. In addition to the scrollbars, mousewheel can be used to scroll the diagram. Diagram's [scrollSettings](#) enable you to read the current scroll status, view port size, current zoom, and zoom factor. It also allows you to scroll the diagram programmatically.

Get current scroll status

Scroll settings allow you to read the scroll status, [viewPortWidth](#), [viewPortHeight](#), and [currentZoom](#) with a set of properties. To explore those properties, see [Scroll Settings](#).

Define scroll status

Diagram allows you to pan the diagram before loading, so that any desired region of a large diagram is made to view. You can programmatically pan the diagram with the [horizontalOffset](#) and [verticalOffset](#) properties of scroll settings. The following code illustrates how to set pan the diagram programmatically.

In the following example, the vertical scroll bar is scrolled down by 50 px and horizontal scroll bar is scrolled to right by 100 px.

INDEX.JS

```
var diagram = new ej.diagrams.Diagram({
  width: '100%',
  height: '700px', scrollSettings: {
    horizontalOffset: 100, verticalOffset: 50
  }
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
```

```
<body>

  <div id="container">
    <div id="element"></div>
  </div>
</body>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Update scroll status

You can programmatically change the scroll offsets at runtime by using the client-side method update. The following code illustrates how to change the scroll offsets and zoom factor at runtime.

INDEX.JS

```
var diagram = new ej.diagrams.Diagram({
  width: '100%',
  height: '700px', scrollSettings: {
    horizontalOffset:100, verticalOffset:50
  }
}, '#element');
//Updates scroll settings
diagram.scrollSettings.horizontalOffset=200;
diagram.scrollSettings.verticalOffset=30
diagram.dataBind();
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

AutoScroll

Autoscroll feature automatically scrolls the diagram, whenever the node or connector is moved beyond the boundary of the diagram. So that, it is always visible during dragging, resizing, and multiple selection operations. Autoscroll is automatically triggered when any one of the following is done towards the edges of the diagram.

- Node dragging, resizing
- Connection editing
- Rubber band selection
- Label dragging

The diagram client-side event [ScrollChange](#) gets triggered when the autoscroll (scrollbars) is changed and you can do your own customization in this event.

The autoscroll behavior in your diagram can be enabled/disabled by using the [canAutoScroll](#) property of the diagram. The following code example illustrates how to set autoscroll.

INDEX.JS

```

var nodes = [{
    id: 'node1', width: 100, height: 100, offsetX: 100, offsetY: 100,
    annotations: [{ content: 'Start' }]
}];
var connectors = [{
    id: 'connector1', sourcePoint: { x: 300, y: 100 }, targetPoint: { x:
500, y: 200 },
    style: {
        strokeColor: '#6BA5D7',
        strokeWidth: 2
    },
    targetDecorator: {
        style: {
            fill: '#6BA5D7',
            strokeColor: '#6BA5D7'
        }
    }
}];

```

```

    },
  }];
  var diagram = new ej.diagrams.Diagram({
    width: '100%',
    height: '600px',
    nodes: nodes,
    connectors: connectors,
    // set the scroll settings
    scrollSettings: { canAutoScroll: true, scrollLimit: 'Infinity' },
    getNodeDefaults: (node) => {
      node.height = 100;
      node.width = 100;
      node.style.fill = '#6BA5D7';
      node.style.strokeColor = 'white';
      return node;
    }
  }, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
  var ele = document.getElementById('container');
  if(ele) {
    ele.style.visibility = "visible";
  }

```

```

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Autoscroll border

The autoscroll border is used to specify the maximum distance between the object and diagram edge to trigger autoscroll. The default value is set as 15 for all sides (left, right, top, and bottom) and it can be changed by using the [autoScrollBorder](#) property of page settings. The following code example illustrates how to set autoscroll border.

INDEX.JS

```

var nodes = [{
  id: 'Start',
  width: 140,
  height: 50,
  offsetX: 300,
  offsetY: 50,
  annotations: [{
    id: 'labell',
    content: 'Start'
  }],
  shape: {
    type: 'Flow',
    shape: 'Terminator'
  }
}];
var diagram = new ej.diagrams.Diagram({
  width: '100%',
  height: '600px',
  nodes: nodes,
  // set the autoScrollBorder
  scrollSettings: { canAutoScroll:
true, autoScrollBorder: { left: 100, right: 100, top: 100, bottom: 100 } },
  getNodeDefaults: (node) => {
    node.height = 100;
    node.width = 100;
    node.style.fill = '#6BA5D7';
    node.style.strokeColor = 'white';
    return node;
  },
  '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">

```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Scroll limit

The scroll limit allows you to define the scrollable region of the diagram. It includes the following options:

- Allows to scroll in all directions without any restriction.
- Allows to scroll within the diagram content.
- Allows to scroll within the specified scrollable area.
- The [scrollLimit](#) property of scroll settings helps to limit the scrolling.

The scrollSettings [scrollableArea](#) allow to extend the scrollable region that is based on the scroll limit.

The following code example illustrates how to specify the scroll limit.

INDEX.JS

```

var nodes = [{
    id: 'Start',
    width: 140,
    height: 50,
    offsetX: 300,
    offsetY: 50,
    annotations: [{
        id: 'labell1',
        content: 'Start'
    }],
}],

```

```

    shape: {
        type: 'Flow',
        shape: 'Terminator'
    }
}];
var diagram = new ej.diagrams.Diagram({
    width: '100%',
    height: '600px',
    nodes: nodes,
    // set the autoScrollBorder
    scrollSettings:{
        canAutoScroll: true,
        //Sets the scroll limit
        scrollLimit: 'infinity'
    },
    getNodeDefaults: (node) => {
        node.height = 100;
        node.width = 100;
        node.style.fill = '#6BA5D7';
        node.style.strokeColor = 'white';
        return node;
    }
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Diagram</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>

```

```

    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Scroll Padding

The [padding](#) property of scroll settings allows you to extend the scrollable region that is based on the scroll limit.

The following code example illustrates how to set scroll padding to diagram region.

INDEX.JS

```

var nodes = [{
    id: 'Start',
    visible: true,
    backgroundColor: 'black',
    offset: 0, side: 'Right', margin: { top: 0, bottom: 0, left: 25, right:
0 },
    shape: {
        type: 'Flow',
        shape: 'Terminator'
    }
}];
var diagram = new ej.diagrams.Diagram({
    width: '100%',
    height: '600px',
    nodes: nodes,
    scrollSettings: {
        canAutoScroll: true,
        //Sets the scroll padding
        padding: { right: 50, bottom: 50 }
    }
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Diagram</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-diagrams/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Scrollable Area

Scrolling beyond any particular rectangular area can be restricted by using the [scrollableArea](#) property of scroll settings. To restrict scrolling beyond any custom region, set the [scrollLimit](#) as “limited”. The following code example illustrates how to customize scrollable area.

INDEX.JS

```

var nodes = [{
    id: 'Start',
    width: 140,
    height: 50,
    offsetX: 300,
    offsetY: 50,
    annotations: [{
        id: 'label1',
        content: 'Start'
    }],
    shape: {
        type: 'Flow',
        shape: 'Terminator'
    }
}];
var diagram = new ej.diagrams.Diagram({
    width: '100%',
    height: '600px',
    nodes: nodes,
    scrollSettings: {
        canAutoScroll: true,
        //Sets the scroll limit
        scrollLimit: 'infinity',
    }
});

```

```

//Sets the scrollable Area
scrollableArea: {
    x: 0,
    y: 0,
    width: 500,
    height: 500
},
getNodeDefaults: (node) => {
    node.height = 100;
    node.width = 100;
    node.style.fill = '#6BA5D7';
    node.style.strokeColor = 'white';
    return node;
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>

```

```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

UpdateViewport

The [updateViewPort](#) method is used to update the diagram page and view size at runtime.

Accessibility in ##Platform_Name## Diagram control

Diagram provides built-in compliance with the [WAI-ARIA](#) specifications. WAI-ARIA Accessibility supports are achieved through the attributes like [aria-label](#). It helps to provides information about elements in a document for assistive technology.

The accessibility compliance for the diagram component is outlined below.

Accessibility Criteria	Compatibility
-----	-----
WCAG 2.2 Support	
Section 508 Support	
Screen Reader Support	
Right-To-Left Support	
Color Contrast	
Mobile Device Support	
Keyboard Navigation Support	
Accessibility Checker Validation	
Axe-core Accessibility Validation	
<style> .post .post-content img { display: inline-block; margin: 0.5em 0; } </style>	

<div> - All features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>

WAI-ARIA attributes

The Diagram component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Diagram component:

| Attributes | Purpose |

| --- | --- |

| [aria-label](#) | Provides an accessible name for the Diagram Objects. |

Aria-label

Attribute provides the text label with some default description for below elements in diagram.

<!-- markdownlint-disable MD033 -->

Element	Default description
ResizeNorthWest	Thumb to resize the selected object on the top-left corner.
ResizeNorthEast	Thumb to resize the selected object on the top-right side direction.
ResizeSouthWest	Thumb to resize the selected object on the bottom-left side direction.
ResizeSouthEast	Thumb to resize the selected object on the bottom-right side direction.
ResizeNorth	Thumb to resize the selected object on the top side direction.
ResizeSouth	Thumb to resize the selected object on the bottom side direction.
ResizeWest	Thumb to resize the selected object on the left side direction.
ResizeEast	Thumb to resize the selected object on the right side direction.
ConnectorSourceThumb	Thumb to move the source point of the connector.
ConnectorTargetThumb	Thumb to move the target point of the connector.
RotateThumb	Thumb to rotate the selected object.

Mobile device support

Syncfusion Diagram component are more user-friendly and accessible to individuals using mobile devices, including those with disabilities. These are designed to be responsive, adaptable to various screen sizes and orientations, and touch-friendly.

Screen Reader Support

The Diagram component supports and its information was dictated properly by the screen readers based on the ARIA attributes and content.

Keyboard navigation support

Syncfusion Diagram component support keyboard navigation, allowing users who rely on alternate methods to effortlessly navigate and interact with the component.

Keyboard interaction

The Diagram component followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Diagram component.

| **Command** | **Action** |

| --- | --- |

| Ctrl + A | Select All |

| Ctrl + X | Cut |

| Ctrl + C | Copy |

| Ctrl + V | Paste |

| Ctrl + Z | Undo |

| Ctrl + Y | Redo |

| Delete | Delete |

| Up Arrow | Move selected object to up |

| Down Arrow | Move selected object to down |

| Left Arrow | Move selected object to left |

| Right Arrow | Move selected object to right |

| Enter | Start Annotation Edit |

| Escape | End Annotation Edit |

Ensuring accessibility

The Diagram component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Diagram component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Diagram component with accessibility tools.

See also

- [Accessibility in Syncfusion Javascript components](#)

Tool tip in ##Platform_Name## Diagram control

<!-- markdownlint-disable MD010 -->

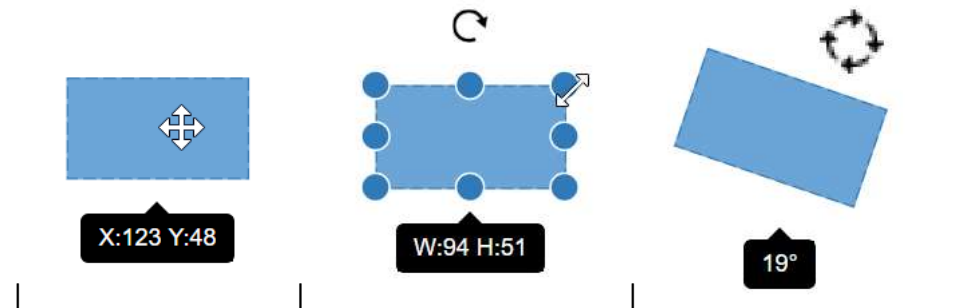
In Graphical User Interface (GUI), the tooltip is a message that is displayed when mouse hovers over an element. The diagram provides tooltip support while dragging, resizing, rotating a node, and when the mouse hovers any diagram element.

Default tooltip

By default, diagram displays a tooltip to provide the size, position, and angle related information while dragging, resizing, and rotating. The following images illustrate how the diagram displays the node information during an interaction.

| Drag | Resize | Rotate |

|---|---|---|



Common tooltip for all nodes and connectors

The diagram provides support to show tooltip when the mouse hovers over any node/connector. To show tooltip on mouse over, the [tooltip](#) property of diagram model needs to be set with the tooltip [content](#) and [position](#) as shown in the following example.

INDEX.JS

```
/**
 * Tooltip sample
 */
var diagram;
var node = {
  id: "node1",
  width: 100,
  height: 100,
  annotations: [{
    id: 'label',
    content: 'Rectangle',
    offset: {
      x: 0.5,
      y: 0.5
    },
    style: {
      color: 'white'
    }
  }],
  offsetX: 200,
  offsetY: 200,
  style: {
    strokeColor: '#6BA5D7',
    fill: '#6BA5D7'
  },
  constraints: ej.diagrams.NodeConstraints.Default |
ej.diagrams.NodeConstraints.Tooltip,
};
diagram = new ej.diagrams.Diagram({
```

```

width: '650px',
height: '350px',
constraints: ej.diagrams.DiagramConstraints.Default |
ej.diagrams.DiagramConstraints.Tooltip,
nodes: [node],
//Defines mouse over tooltip
tooltip: {
  content: 'Nodes',
  position: 'TopLeft'
}
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  popups/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  diagrams/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  navigations/styles/fabric.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Disable tooltip at runtime

The tooltip on mouse over can be disabled by assigning the [tooltip](#) property as `null`. The following code example illustrates how to disable the mouse over tooltip at runtime.

```
`ts
//Initializes the diagram component
let diagram: Diagram = new Diagram({
width: '100%', height: '350px',
//Disables mouse over tooltip at runtime
tooltip: null
}, '#element');
`
```

Tooltip for a specific node/connector

The tooltip can be customized for each node and connector. Remove the **InheritTooltip** option from the [constraints](#) of that node/connector. The following code example illustrates how to customize the tooltip for individual elements.

INDEX.JS

```
/**
 * Tooltip sample
 */
var diagram;
var node = {
  id: "node1",
  width: 100,
  height: 100,
  annotations: [{
    id: 'label',
    content: 'Rectangle',
    offset: {
      x: 0.5,
      y: 0.5
    },
    style: {
      color: 'white'
    }
  }],
  offsetX: 200,
  offsetY: 200,
  style: {
    strokeColor: '#6BA5D7',
    fill: '#6BA5D7'
  },
  constraints: ej.diagrams.NodeConstraints.Default |
ej.diagrams.NodeConstraints.Tooltip,
  //Defines mouse over tooltip for a node
  tooltip: {
    //Sets the content of the Tooltip
    content: 'Node1',
    //Sets the position of the Tooltip
  }
}
```

```

        position: 'BottomRight',
        //Sets the tooltip position relative to the node
        relativeMode: 'Object'
    },
};
diagram = new ej.diagrams.Diagram({
    width: '650px',
    height: '350px',
    constraints: ej.diagrams.DiagramConstraints.Default |
ej.diagrams.DiagramConstraints.Tooltip,
    nodes: [node],
}, '#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Diagram</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Tooltip for Ports

The tooltip feature has been implemented to support Ports, providing the ability to display information or descriptions when the mouse hovers over them.

To display tooltips on mouseover, set the desired tooltip [content](#) by utilizing the [tooltip](#) property.

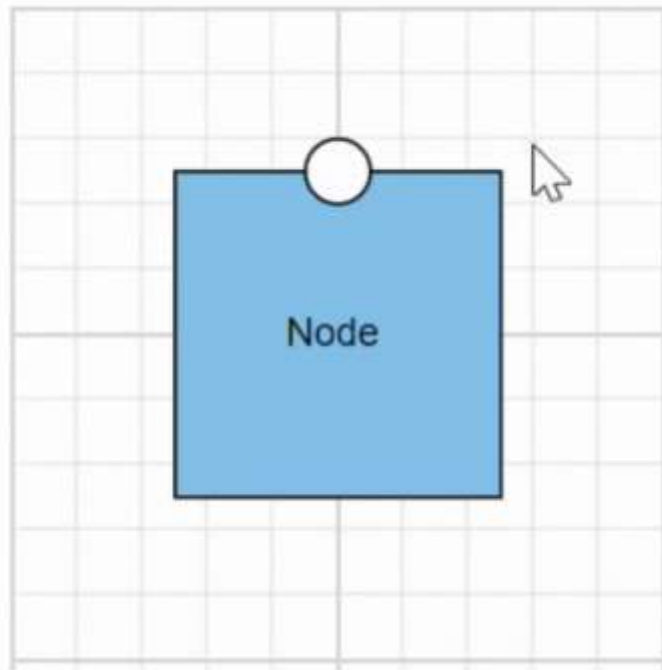
Tooltips for Ports can be enabled or disabled using the [PortConstraints](#) Tooltip property.

```
`ts
let ports: [{
  offset: {x: 1,y: 0.5},
  tooltip: {content: 'Port Tootip'},
  //enable Port Tooltip Constraints
  constraints: PortConstraints.Default | PortConstraints.ToolTip,
  //disable Port Tooltip Constraints
  constraints: PortConstraints.Default ~& PortConstraints.ToolTip
}]
`
```

Dynamic modification of tooltip content is supported, allowing you to change the displayed tooltip content during runtime.

```
`ts
{
  //change tooltip content at run time
  diagram.nodes[0].ports[0].tooltip.content = 'New Tooltip Content';
  diagram.databind;
}
`
```

The following image illustrates how the diagram displays tooltips during an interaction with ports:



Here, the code provided below demonstrates the port tooltip Interaction.

INDEX.JS

```

/**Tooltip sample */
var node = {
  // Position of the node
  offsetX: 250,
  offsetY: 250,
  // Size of the node
  width: 100,
  height: 100,
  style: { fill: '#6BA5D7', strokeColor: 'white' },
  ports: [{
    offset: { x: 0.5, y: 0.5 }, visibility:
ej.diagrams.PortVisibility.Visible,
    constraints: ej.diagrams.PortConstraints.Default |
ej.diagrams.PortConstraints.Tooltip,
    //Defines mouse over tooltip for a node
    tooltip: {
      content: 'Port Tooltip',
    },
  ]
};
// initialize Diagram component
var diagram = new ej.diagrams.Diagram({
  width: '100%', height: '600px', nodes: [node]
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>

```

```

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Tooltip template content

Any text or image can be added to the tooltip, by default. To customize the tooltip layout or to create your own visualized element on the tooltip, template can be used.

The following code example illustrates how to add formatted HTML content to the tooltip.

INDEX.JS

```

/**
 * Tooltip sample
 */
var diagram;
var node = {
    id: "node1",
    width: 100,
    height: 100,

```

```

    annotations: [{
      id: 'label',
      content: 'Rectangle',
      offset: {
        x: 0.5,
        y: 0.5
      },
      style: {
        color: 'white'
      }
    }],
    offsetX: 200,
    offsetY: 200,
    style: {
      strokeColor: '#6BA5D7',
      fill: '#6BA5D7'
    },
    constraints: ej.diagrams.NodeConstraints.Default |
ej.diagrams.NodeConstraints.Tooltip,
    //Defines mouse over tooltip for a node
    tooltip: {
      //Sets the content of the Tooltip
      content: getContent(),
      //Sets the position of the Tooltip
      position: 'TopLeft',
      //Sets the tooltip position relative to the node
      relativeMode: 'Object'
    }
  };
  diagram = new ej.diagrams.Diagram({
    width: '100%',
    height: '350px',
    constraints: ej.diagrams.DiagramConstraints.Default |
ej.diagrams.DiagramConstraints.Tooltip,
    nodes: [node],
  }, '#element');
  function getContent() {
    var tooltipContent = document.createElement('div');
    tooltipContent.innerHTML = '<div> <rect style="background-color:
#f4f4f4; color: black; border-width:1px;border-style: solid;border-color:
#d3d3d3; border-radius: 8px;white-space: nowrap;"> Tooltip !!! </rect>
</div>';
    return tooltipContent;
  }

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Tooltip alignments

Tooltip relative to object

The diagram provides support to show tooltip around the node/connector that is hovered by the mouse. The tooltip can be aligned by using the [position](#) property of the tooltip. The [relativeMode](#) property of the tooltip defines whether the tooltip has to be displayed around the object or at the mouse position.

The following code example illustrates how to position the tooltip around object.

INDEX.JS

```

/**
 * Tooltip sample
 */
var diagram;
var node = {
    id: "node1",
    width: 100,
    height: 100,
    annotations: [{
        id: 'label',
        content: 'Rectangle',
        offset: {
            x: 0.5,

```

```

        y: 0.5
      },
      style: {
        color: 'white'
      }
    }],
    offsetX: 200,
    offsetY: 200,
    style: {
      strokeColor: '#6BA5D7',
      fill: '#6BA5D7'
    },
    constraints: (ej.diagrams.NodeConstraints.Default &
~ej.diagrams.NodeConstraints.InheritTooltip) |
ej.diagrams.NodeConstraints.Tooltip,
    //Defines mouse over tooltip for a node
    tooltip: {
      //Sets the content of the Tooltip
      content: 'Node1',
      //Sets the position of the Tooltip
      position: 'BottomRight',
      //Sets the tooltip position relative to the node
      relativeMode: 'Object'
    },
  },
};
diagram = new ej.diagrams.Diagram({
  width: '100%',
  height: '350px',
  constraints: ej.diagrams.DiagramConstraints.Default |
ej.diagrams.DiagramConstraints.Tooltip,
  nodes: [node],
}, '#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Tooltip relative to mouse position

To display the tooltip at mouse position, need to set **mouse** option to the [relativeMode](#) property of the tooltip.

The following code example illustrates how to show tooltip at mouse position.

INDEX.JS

```

/**
 * Tooltip sample
 */
var diagram;
var node = {
    id: "node1",
    width: 100,
    height: 100,
    annotations: [{
        id: 'label',
        content: 'Rectangle',
        offset: {
            x: 0.5,
            y: 0.5
        },
        style: {
            color: 'white'
        }
    }],
    offsetX: 200,
    offsetY: 200,
    style: {
        strokeColor: '#6BA5D7',
        fill: '#6BA5D7'
    },
},

```

```

    constraints: ej.diagrams.NodeConstraints.Default |
ej.diagrams.NodeConstraints.Tooltip,
    //Defines mouse over tooltip for a node
    tooltip: {
        //Sets the content of the Tooltip
        content: 'Node1',
        //Sets the tooltip position relative to the node
        relativeMode: 'Mouse'
    },
};
diagram = new ej.diagrams.Diagram({
    width: '650px',
    height: '350px',
    constraints: ej.diagrams.DiagramConstraints.Default |
ej.diagrams.DiagramConstraints.Tooltip,
    nodes: [node],
}, '#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Diagram</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
```

```

    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Tooltip animation

To animate the tooltip, a set of specific animation effects are available, and it can be controlled by using the [animation](#) property. The animation property also allows you to set delay, duration, and various other effects of your choice.

INDEX.JS

```

/**
 * Tooltip sample
 */
var diagram;
diagram = new ej.diagrams.Diagram({
    width: '650px',
    height: '350px',
    constraints: ej.diagrams.DiagramConstraints.Default |
ej.diagrams.DiagramConstraints.Tooltip,
    //Defines nodes
    nodes: [{
        id: "node1",
        width: 100,
        height: 100,
        annotations: [{
            id: 'label',
            content: 'Rectangle',
            offset: {
                x: 0.5,
                y: 0.5
            },
            style: {
                color: 'white'
            }
        }],
        offsetX: 200,
        offsetY: 200,
        style: {
            strokeColor: '#6BA5D7',
            fill: '#6BA5D7'
        },
        constraints: ej.diagrams.NodeConstraints.Default |
ej.diagrams.NodeConstraints.Tooltip,
        //Defines mouse over tooltip for a node
        tooltip: {
            content: 'Node1',
            position: 'BottomCenter',
            relativeMode: 'Object',
            animation: {
                //Animation settings to be applied on the Tooltip, while it
                is being shown over the target.
                open: {

```

```
//Animation effect on the Tooltip is applied during open
and close actions.
effect: 'FadeIn',
//Duration of the animation that is completed per
animation cycle.
duration: 150,
//Indicating the waiting time before animation begins.
delay: 0
},
//Animation settings to be applied on the Tooltip, when it
is closed.
close: {
    effect: 'FadeOut',
    delay: 0
}
},
}]
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
</script>
```

```

var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

User handle in ##Platform_Name## Diagram control

- User handles are used to add some frequently used commands around the selector. To create user handles, define and add them to the [userHandles](#) collection of the [selectedItems](#) property.
- The name property of user handle is used to define the name of the user handle and its further used to find the user handle at runtime and do any customization.

Alignment

User handles can be aligned relative to the node boundaries. It has [margin](#), [offset](#), [side](#), [horizontalAlignment](#), and [verticalAlignment](#) settings. It is quite tricky when all four alignments are used together but gives more control over alignment.

Offset for user handle

The [offset](#) property of [userHandles](#) is used to align the user handle based on fractions. 0 represents top/left corner, 1 represents bottom/right corner, and 0.5 represents half of width/height.

Side

The [side](#) property of [userHandles](#) is used to align the user handle by using the [Top](#), [Bottom](#), [Left](#), and [Right](#) options.

Horizontal and vertical alignments

The [horizontalAlignment](#) property of [userHandles](#) is used to set how the user handle is horizontally aligned at the position based on the [offset](#). The [verticalAlignment](#) property is used to set how user handle is vertically aligned at the position.

Margin for the user handle

Margin is an absolute value used to add some blank space in any one of its four sides. The [userHandles](#) can be displaced with the [margin](#) property.

Appearance

The appearance of the user handle can be customized by using the [size](#), [borderColor](#), [backgroundColor](#), [visible](#), [pathData](#), and [pathColor](#) properties of the [userHandles](#).

INDEX.JS

```

var shape = { type: 'Basic', shape: 'Rectangle' };
var node1 = { id: 'node', offsetX: 100, offsetY: 100, shape: shape };
var handles = [{
  name: 'clone', pathData: 'M60.3,18H27.5c-3,0-5.5,2.4-5.5,5.5v38.2h5.5V23.5h32.7V18z M68.5,28.9h-30c-3,0-5.5,2.4-5.5,5.5v38.2c0,3,2.4,5.5,5.5,5.5h30c3,0,5.5-2.4,5.5-5.5V34.4C73.9,31.4,71.5,28.9,68.5,28.9z ' +
  'M68.5,72.5h-30V34.4h30V72.5z',
  visible: true, offset: 0, side: 'Bottom', margin: { top: 0, bottom: 0, left: 0, right: 0 }
}];

```

```

    }];
    //Defines the diagram content
    diagram = new ej.diagrams.Diagram({
        width: '100%', height: '600px',
        nodes: [node1],
        selectedItems: { constraints: ej.diagrams.SelectorConstraints.All,
userHandles: handles },
        getNodeDefaults: function (node) {
            node.height = 100;
            node.width = 100;
            node.style.fill = '#6BA5D7';
            node.style.strokeColor = 'white';
            return node;
        },
        getCustomTool: getTool
    });
    diagram.appendTo('#element');
    //Enable the clone Tool for UserHandle.
    function getTool(action) {
        var tool;
        if (action === 'clone') {
            tool = new CloneTool(diagram.commandHandler);
        }
        return tool;
    }
    var __extends = (this && this.__extends) || (function () {
    var extendStatics = Object.setPrototypeOf ||
        /* jshint proto: true */
        ({ __proto__: [] } instanceof Array && function (d, b) { d.__proto__
= b; }) ||
        function (d, b) { for (var p in b) if (b.hasOwnProperty(p)) d[p] =
b[p]; };
    return function (d, b) {
        extendStatics(d, b);
        function __() { this.constructor = d; }
        d.prototype = b === null ? Object.create(b) : (__.prototype =
b.prototype, new __());
    };
    })();
    var CloneTool = (function (_super) {
        __extends(CloneTool, _super);
        function CloneTool() {
            return _super !== null && _super.apply(this, arguments) || this;
        }
        CloneTool.prototype.mouseDown = function (args) {
            var newObject;
            if (diagram.selectedItems.nodes.length > 0) {
                newObject =
ej.diagrams.cloneObject(diagram.selectedItems.nodes[0]);
            }
            else {
                newObject =
ej.diagrams.cloneObject(diagram.selectedItems.connectors[0]);
            }
            newObject.id += ej.diagrams.randomId();
            diagram.paste([newObject]);
            args.source = diagram.nodes[diagram.nodes.length - 1];
        }
    })(CloneTool);

```



```

        args.sourceWrapper = args.source.wrapper;
        _super.prototype.mouseDown.call(this, args);
        this.inAction = true;
    };
    return CloneTool;
}(ej.diagrams.MoveTool));

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Fixed user handles

The fixed user handles are used to add some frequently used commands around the node and connector even without selecting it.

Initialization an fixed user handles

To create the fixed user handles, define and add them to the collection of nodes and connectors property. The following code example used to create an fixed user handles for the nodes and connectors.

INDEX.JS

```
var node = {
  // Position of the node
  offsetX: 250,
  offsetY: 250,
  // Size of the node
  width: 100,
  height: 100,
  style: { fill: '#6BA5D7', strokeColor: 'white' },
  fixedUserHandles: [{ margin: { right: 20 }, pathData: 'M60.3,18H27.5c-
3,0-5.5,2.4-5.5,5.5v38.2h5.5V23.5h32.7V18z M68.5,28.9h-30c-3,0-5.5,2.4-
5.5,5.5v38.2c0,3,2.4,5.5,5.5,5.5h30c3,0,5.5-2.4,5.5-
5.5V34.4C73.9,31.4,71.5,28.9,68.5,28.9z M68.5,72.5h-30V34.4h30V72.5z' }]
};
// initialize Diagram component
var diagram = new ej.diagrams.Diagram({
  width: '100%', height: '600px', nodes: [node]
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
```

```

        <div id="element"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customization

- The id property of fixed user handle is used to define the unique identification of the fixed user handle and it is further used to add custom events to the fixed user handle.
- The fixed user handle can be positioned relative to the node and connector boundaries. It has offset, padding and cornerRadius settings. It is used to position and customize the fixed user handle.
- The **Padding** is used to leave the space that is inside the fixed user handle between the icon and border.
- The corner radius allows to create fixed user handles with rounded corners. The radius of the rounded corner is set with the **cornerRadius** property.

Note: The PathData needs to be provided to render fixed user handle.

Size

Diagram allows you set size for the fixed user handles by using the **width** and **height** property. The default value of the width and height property is 10.

Style

- You can change the style of the fixed user handles with the specific properties of **borderColor**, **borderWidth**, and background color using the **handleStrokeColor**, **handleStrokeWidth**, and **fill** properties, and the icon **borderColor**, and **borderWidth** using the **iconStrokeColor** and **iconStrokeWidth**.
- The fixed user handle's **iconStrokeColor** and **iconStrokeWidth** property used to change the stroke color and stroke width of the given **pathData**.
- The fixed user handle **handleStrokeColor** and **fill** properties are used to define the background color and border color of the userhandle and the **handleStrokeWidth** property is used to define the border width of the fixed user handle.
- The **visible** property of the fixed user handle enables or disables the visibility of fixed user handle.

The following code explains how to customize the appearance of the fixed user handles.

INDEX.JS

```

var nodes = {
    id: 'node1', width: 100, height: 100, offsetX: 100, offsetY:
    150, isExpanded: true,

```

```

        fixedUserHandles: [{ offset: { x: 0, y: 0 }, margin: { right: 20
    }, fill: 'black', iconStrokeColor: 'white', handleStrokeWidth: 3, handleStrokeColor
    : 'black', width: 20, height: 20, id: 'user1', pathData: 'M60.3,18H27.5c-
    3,0-5.5,2.4-5.5,5.5v38.2h5.5V23.5h32.7V18z M68.5,28.9h-30c-3,0-5.5,2.4-
    5.5,5.5v38.2c0,3,2.4,5.5,5.5,5.5h30c3,0,5.5-2.4,5.5-
    5.5V34.4C73.9,31.4,71.5,28.9,68.5,28.9z M68.5,72.5h-30V34.4h30V72.5z' }]
    };
    var connector = {
        id: 'connector1',
        type: 'Orthogonal',
        sourcePoint: { x: 300, y: 100 },
        targetPoint: { x: 400, y: 200 }, fixedUserHandles: [{ offset:
0.5, width:
20, fill: 'black', iconStrokeColor: 'white', handleStrokeWidth: 3, handleStrokeColor
: 'black', alignment: 'Before', height: 20, id: 'usercon1', displacement: {
x: 10, y: 10 }, pathData: 'M60.3,18H27.5c-3,0-5.5,2.4-
5.5,5.5v38.2h5.5V23.5h32.7V18z M68.5,28.9h-30c-3,0-5.5,2.4-
5.5,5.5v38.2c0,3,2.4,5.5,5.5,5.5h30c3,0,5.5-2.4,5.5-
5.5V34.4C73.9,31.4,71.5,28.9,68.5,28.9z M68.5,72.5h-30V34.4h30V72.5z' }]
    };
    var diagram = new ej.diagrams.Diagram({
        width: '100%', height: '600px', nodes: [nodes], connectors: [connector]
    }, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Diagram</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
    base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
    buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
    popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
    splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
    diagrams/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
    navigations/styles/fabric.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>

```

```

</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Note: The fixed user handle id need to be unique.

Customizing the node fixed user handle

- The node fixed user handle can be aligned relative to the node boundaries. It has **margin** and **offset** settings. It is quite useful to position the node fixed userhandle and used together and gives you more control over the node fixed user handle positioning.

Margin for the node fixed user handle

Margin is an absolute value used to add some blank space in any one of its four sides. The fixed user handle can be displaced with the **margin** property.

Offset for the node fixed user handle

The **offset** property of fixed user handle is used to align the user handle based on the **x** and **y** points. (0,0) represents the top or left corner and (1,1) represents the bottom or right corner.

The following table shows all the possible alignments visually shows the fixed user handle positions.

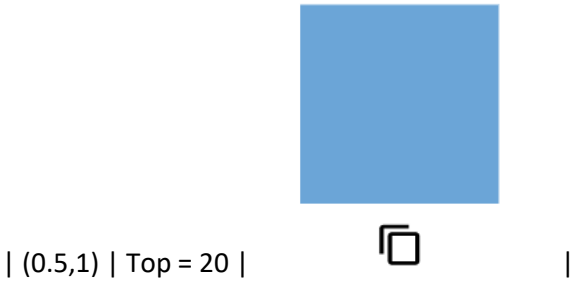
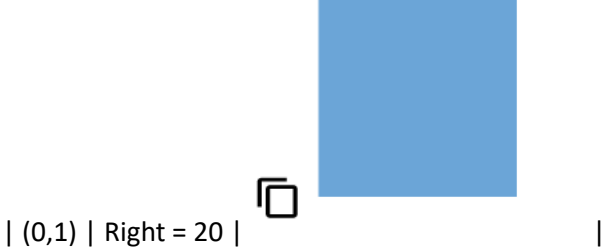
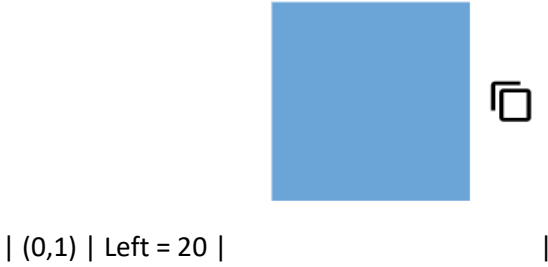
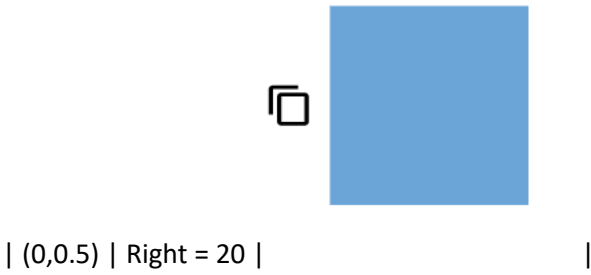
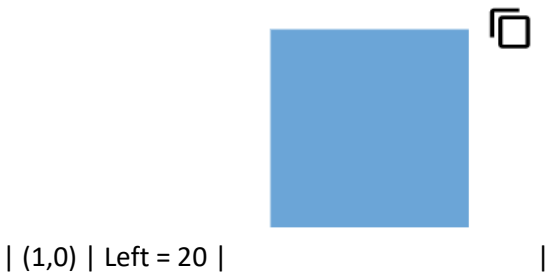
Offset	Margin	Output
(0,0)	Right = 20	
(0.5,0)	Bottom = 20	



| (0,0) | Right = 20 |



| (0.5,0) | Bottom = 20 |





| (1,1) | Left = 20 |

The following code explains how to customize the node fixed user handle.

INDEX.JS

```
var node = {
  // Position of the node
  offsetX: 250,
  offsetY: 250,
  // Size of the node
  width: 100,
  height: 100,
  style: { fill: '#6BA5D7', strokeColor: 'white' },
  fixedUserHandles: [{ offset: { x: 0, y: 0 }, margin: { right: 20 },
width: 20,height: 20, id: 'user1', pathData: 'M60.3,18H27.5c-3,0-5.5,2.4-
5.5,5.5v38.2h5.5V23.5h32.7V18z M68.5,28.9h-30c-3,0-5.5,2.4-
5.5,5.5v38.2c0,3,2.4,5.5,5.5,5.5h30c3,0,5.5-2.4,5.5-
5.5V34.4C73.9,31.4,71.5,28.9,68.5,28.9z M68.5,72.5h-30V34.4h30V72.5z' }]
  };
// initialize Diagram component
var diagram = new ej.diagrams.Diagram({
  width: '100%', height: '600px', nodes: [node]
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customizing the connector fixed user handle

- The connector fixed user handle can be aligned relative to the connector boundaries. It has alignment, displacement and offset settings. It is useful to position the connector fixed user handle and used together and gives you more control over the connector fixed user handle positioning.
- The **offset** and **alignment** properties of fixed user handle allows you to align the connector fixed user handles to the segments.

Offset for the connector fixed user handle

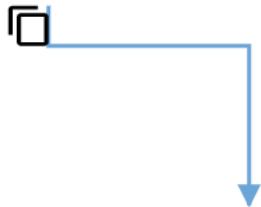
The **offset** property of connector fixed user handle is used to align the user handle based on fractions. 0 represents the connector source point, 1 represents the connector target point, and 0.5 represents the center point of the connector segment.

Alignment

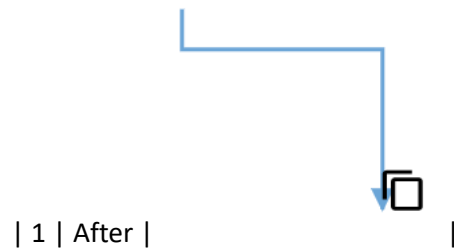
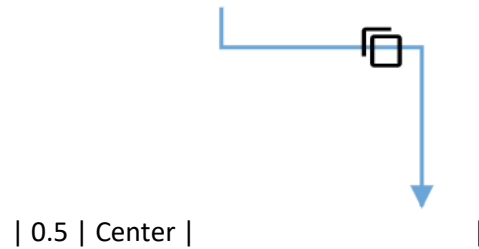
The connector's fixed user handle can be aligned over its segment path using the **alignment** property of fixed user handle.

The following table shows all the possible alignments visually shows the fixed user handle positions.

Offset	Alignment	Output
-----	-----	-----



0 Before	
------------	--

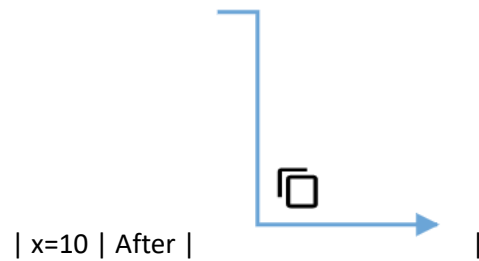
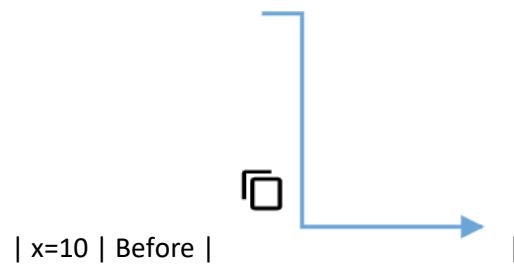


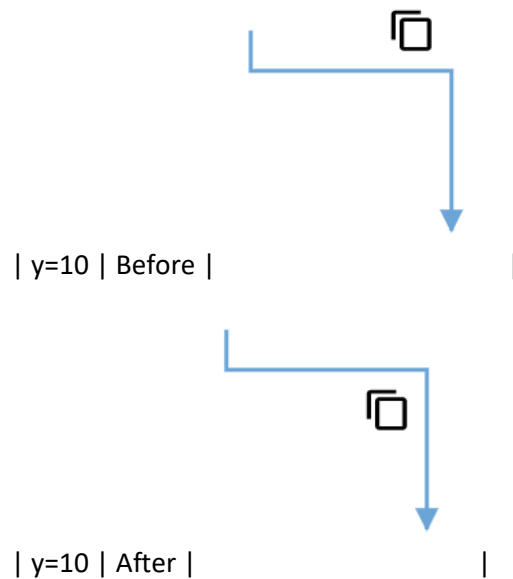
Displacement

- The **displacement** property allows you to specify the space to be left from the connector segment based on the x and y value provided.

The following table shows all the possible alignments visually shows the fixed user handle positions.

Displacement	Alignment	Output
-----	-----	-----





Note: Displacement will not be done if the alignment is set to be center.

The following code explains how to customize the connector fixed user handle.

INDEX.JS

```
var connector = {
  id: 'connector1',
  type: 'Orthogonal',
  sourcePoint: { x: 300, y: 100 },
  targetPoint: { x: 400, y: 200 }, fixedUserHandles: [{ offset:
0.5, width: 20, alignment: 'Before', height: 20, id: 'usercon1',
displacement: { x: 10, y: 10 }, pathData: 'M60.3,18H27.5c-3,0-5.5,2.4-
5.5,5.5v38.2h5.5V23.5h32.7V18z M68.5,28.9h-30c-3,0-5.5,2.4-
5.5,5.5v38.2c0,3,2.4,5.5,5.5,5.5h30c3,0,5.5-2.4,5.5-
5.5V34.4C73.9,31.4,71.5,28.9,68.5,28.9z M68.5,72.5h-30V34.4h30V72.5z' }]
};
var diagram = new ej.diagrams.Diagram({
  width: '100%', height: '600px', connectors: [connector]
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Tooltip support for User Handle

The diagram provides support to show tooltip when the mouse hovers over any user handle. To show the tooltip on mouse over, the [tooltip](#) property of diagram model needs to be set with the tooltip [content](#) and [position](#) as shown in the following example.

INDEX.JS

```

var node = {
    // Position of the node
    offsetX: 250,
    offsetY: 250,
    // Size of the node
    width: 100,
    height: 100,
    style: { fill: '#6BA5D7', strokeColor: 'white' },
    tooltip: { content: 'node1', position: 'BottomRight', relativeMode:
'Object' },
    constraints: ej.diagrams.NodeConstraints.Default |
ej.diagrams.NodeConstraints.Tooltip,
};
var handle = [{
    name: 'clone', pathData: 'M60.3,18H27.5c-3,0-5.5,2.4-
5.5,5.5v38.2h5.5V23.5h32.7V18z M68.5,28.9h-30c-3,' +
'0-5.5,2.4-5.5,5.5v38.2c0,3,2.4,5.5,5.5,5.5h30c3,0,5.5-2.4,5.5-
5.5V34.4C73.9,31.4,71.5,28.9,68.5,28.9z ' +
'M68.5,72.5h-30V34.4h30V72.5z',
    visible: true, offset: 0, side: 'Bottom', margin: { top: 0, bottom:
0, left: 0, right: 0 },

```

```

        tooltip: { content: 'handle1', position: 'BottomRight',
relativeMode: 'Object' }
    }];
    // initialize Diagram component
    var diagram = new ej.diagrams.Diagram({
        width: '100%', height: '600px', nodes: [node],
        constraints: ej.diagrams.DiagramConstraints.Default |
ej.diagrams.DiagramConstraints.Tooltip,
        nodeTemplate: '#nodetemplate',
        userHandleTemplate: '#userHandletemplate',
        selectedItems: { constraints:
ej.diagrams.SelectorConstraints.All,userHandles: handle },
    }, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Diagram</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
    <script>
    var ele = document.getElementById('container');
    if(ele) {
        ele.style.visibility = "visible";
    }
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

Style in ##Platform_Name## Diagram control

Customizing the connector end point handle

Use the following CSS to customize the connector end point handle.

```
`scss
.e-diagram-endpoint-handle {
  fill: red;
  stroke: green;
}
`
```

Customizing the connector end point handle when connected

Use the following CSS to customize the connector end point handle when connected.

```
`scss
.e-diagram-endpoint-handle.e-connected {
  fill: red;
  stroke: green;
}
`
```

Customizing the connector end point handle when disabled

Use the following CSS to customize the connector end point handle when disabled.

```
`scss
.e-diagram-endpoint-handle.e-disabled {
  fill: red;
  opacity: 1;
  stroke: green;
}
`
```

Customizing the bezier connector handle

Use the following CSS to customize the bezier handle properties.

```
`scss
.e-diagram-bezier-handle {
  fill: red;
  stroke: green;
}
`
```

Customizing the bezier connector line

Use the following CSS to customize the bezier line properties.

```
`scss
.e-diagram-bezier-line {
stroke: black;
}
`
```

Customizing the resize handle

Use the following CSS to customize the resize handle.

```
`scss
.e-diagram-resize-handle {
fill: white;
opacity: 1;
stroke: white;
}
`
```

Customizing the selector pivot line

Use the following CSS to customize the line between the selector and rotate handle.

```
`scss
.e-diagram-pivot-line {
stroke: red;
}
`
```

Customizing the selector border

Use the following CSS to customize the selector border.

```
`scss
.e-diagram-border {
stroke: red;
}
`
```

Customizing the rotate handle

Use the following CSS to customize the rotate handle properties.

```
`scss
.e-diagram-rotate-handle {
```

```
fill: red;
stroke: green;
}
```

Customizing the symbolpalette while hovering

Use the following CSS to customize the symbolpalette while hovering.

```
`scss
.e-symbolpalette .e-symbol-hover:hover {
background: red;
}
```

Customizing the symbolpalette when selected

Use the following CSS to customize the symbolpalette when selected.

```
`scss
.e-symbolpalette .e-symbol-selected {
background: white;
}
```

Customizing the ruler

Use the following CSS to customize the ruler properties.

```
`scss
.e-diagram .e-ruler {
background-color: red;
font-size: 13px;
}
```

Customizing the ruler overlap

Use the following CSS to ruler overlap properties.

```
`scss
.e-diagram .e-ruler-overlap {
background-color: red;
}
```

Customizing the text edit

Use the following CSS to customize the text edit properties.

```
`scss
.e-diagram .e-diagram-text-edit {
background: white;
border-color: red;
border-style: dashed;
border-width: 1px;
box-sizing: content-box;
color: black;
min-width: 50px;
}
`
```

Customizing the text edit on selection

Use the following CSS to customize the text edit on selection properties.

```
`scss
.e-diagram-text-edit::selection {
background: red;
color: green;
}
`
```

Ruler in ##Platform_Name## Diagram control

The Ruler provides a horizontal and vertical guide for measuring in the Diagram control. The Ruler can be used to measure the diagram objects, indicate positions, and align diagram elements. This is especially useful in creating scale models.

Adding Rulers to the Diagram

- The [rulerSettings](#) property is used to control the visibility and appearance of the ruler in the diagram.
- The RulerSettings [showRulers](#) property is used to show or hide the rulers in the diagram.
- The RulerSettings [horizontalRuler](#) and [verticalRuler](#) properties are used to customize the rulers appearance in the diagram.

The following code shows how to add a ruler to the diagram.

INDEX.JS

```
var diagram = new ej.diagrams.Diagram({
  width: '100%', height: '600px',
  rulerSettings: {
```



```

        showRulers: true
    },
});
diagram.appendTo('#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>

  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Customizing the Ruler

By default, the ruler segments are arranged based on pixel values.

- The `HorizontalRuler`'s [interval](#) property allows you to define the interval between ruler segments and the [segmentWidth](#) property allows you to define the segment width of the ruler. Similarly,

you can use the VerticalRuler's [interval](#) and [segmentWidth](#) properties are used to define the interval and segment width of the vertical ruler

- The HorizontalRuler's [tickAlignment](#) property is used to align the ruler tick either left or right side of the ruler. The VerticalRuler's [tickAlignment](#) property is used to align the ruler tick either top or bottom side of the ruler.
- The HorizontalRuler's [arrangeTick](#) and VerticalRuler's [arrangeTick](#) function is provided for the purpose of customizing the appearance of ruler ticks. It will be called for each tick rendering.
- The HorizontalRuler's [markerColor](#) and VerticalRuler's [markerColor](#) properties are used to define the ruler marker color and marker will be shown when performing the interaction in the diagram.

The following code shows how the diagram ruler can be customized.

INDEX.JS

```
var diagram = new ej.diagrams.Diagram({
  width: '100%', height: '600px',
  rulerSettings: {
    showRulers: true, horizontalRuler:{interval:8, segmentWidth:100,
    thickness:35, tickAlignment:"LeftOrTop"}, verticalRuler:{interval:10,
    segmentWidth:150, thickness:35, tickAlignment:"RightOrBottom"}
  },
});
diagram.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
```

```

<div id="container">
  <div id="element"></div>
</div>

<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Note : The MarkerColor property can be customized using the [marker](#) CSS style.

Context menu in ##Platform_Name## Diagram control

<!-- markdownlint-disable MD010 -->

In graphical user interface (GUI), a context menu is a type of menu that appears when you perform right-click operation. Nested level of context menu items can be created. Diagram provides some in-built context menu items and allows to define custom menu items through the [contextMenuSettings](#) property.

Customize context menu

The [show](#) property helps you to enable/disable the context menu. Diagram provides some default context menu items to ease the execution of some frequently used commands. The following code illustrates how to enable the default context menu items.

INDEX.JS

```

var nodes = [{
  id: 'node1',
  width: 100,
  height: 100,
  offsetX: 100,
  offsetY: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white',
    strokeWidth: 1
  },
  annotations: [{
    id: 'label1',
    content: 'Rectangle1',
    offset: {
      x: 0.5,
      y: 0.5
    },
    style: {
      color: 'white'
    }
  }]
},
{
  id: 'node2',
  width: 100,

```

```

        height: 100,
        offsetX: 300,
        offsetY: 100,
        style: {
            fill: '#6BA5D7',
            strokeColor: 'white',
            strokeWidth: 1
        },
        annotations: [{
            id: 'label2',
            content: 'Rectangle2',
            offset: {
                x: 0.5,
                y: 0.5
            },
            style: {
                color: 'white'
            }
        }]
    },
];
var connector = {
    id: 'connector1',
    sourceID: 'node1',
    targetID: 'node2',
    type: 'Straight',
    style: {
        strokeColor: '#6BA5D7',
        fill: '#6BA5D7',
        strokeWidth: 2,
        targetDecorator: {
            style: {
                fill: '#6BA5D7',
                strokeColor: '#6BA5D7'
            }
        }
    }
};
var diagram = new ej.diagrams.Diagram({
    width: '100%',
    height: '350px',
    nodes: nodes,
    connectors: [connector],
    //Enables the context menu
    contextMenuSettings: {
        show: true,
    }
}, '#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
```

```

<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Context menu can be defined for individual node with the desired context menu items.

- Apart from the default context menu items, define some additional context menu items. Those additional items have to be defined and added to the [items](#) property of the context menu.
- Set text and ID for context menu item using the context menu [text](#) and [ID](#) properties respectively.
- Set an image for the context menu item using the context menu [url](#) property.
- The [iconCss](#) property defines the class/multiple classes separated by a space for the menu item that is used to include an icon. Menu item can include font icon and sprite image.
- The [target](#) property used to set the target to show the menu item.
- The [separator](#) property defines the horizontal lines that are used to separate the menu items. You cannot select the separators. You can enable separators to group the menu items using the separator property.

The following code example illustrates how to add custom context menu items.

INDEX.JS

```
var nodes = [{
```

```

        id: 'node1',
        width: 100,
        height: 100,
        offsetX: 100,
        offsetY: 100,
        style: {
            fill: '#6BA5D7',
            strokeColor: 'white',
            strokeWidth: 1
        },
        annotations: [{
            id: 'label1',
            content: 'Rectangle1',
            offset: {
                x: 0.5,
                y: 0.5
            },
            style: {
                color: 'white'
            }
        }]
    },
    {
        id: 'node2',
        width: 100,
        height: 100,
        offsetX: 300,
        offsetY: 100,
        style: {
            fill: '#6BA5D7',
            strokeColor: 'white',
            strokeWidth: 1
        },
        annotations: [{
            id: 'label2',
            content: 'Rectangle2',
            offset: {
                x: 0.5,
                y: 0.5
            },
            style: {
                color: 'white'
            }
        }]
    },
];
var connector = {
    id: 'connector1',
    sourceID: 'node1',
    targetID: 'node2',
    type: 'Straight',
    style: {
        strokeColor: '#6BA5D7',
        fill: '#6BA5D7',
        strokeWidth: 2,
        targetDecorator: {
            style: {

```

```

        fill : '#6BA5D7',
        strokeColor : '#6BA5D7'
    }
}
};
var diagram = new ej.diagrams.Diagram({
    width: '100%',
    height: '350px',
    nodes: nodes,
    connectors: [connector],
    contextMenuSettings: {
        //Enables the context menu
        show: true,
        // Defines the custom context menu items
        items: [{
            // Text to be displayed
            text: 'Save',
            id: 'save',
            target: '.e-elementcontent',
            // Sets the css icons for the item
            iconCss: 'e-save'
        },
        {
            text: 'Load',
            id: 'load',
            target: '.e-elementcontent',
            iconCss: 'e-load'
        },
        {
            text: 'Clear',
            id: 'clear',
            target: '.e-elementcontent',
            iconCss: 'e-clear'
        }
    ],
    // Hides the default context menu items
    showCustomMenuOnly: false,
}, '#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Diagram</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

To display the custom context menu items alone, set the [showCustomMenuOnly](#) property to true.

Template Support for Context menu

- Diagram provides template support for context menu. The context menu items can be customized by using the `contextMenuBeforeItemRender` event. The `contextMenuBeforeItemRender` event triggers while rendering each menu item.
- In the following sample, the menu item is rendered with key code for specified action in ContextMenu using the template. Here, the key code is specified for the cut and copy at right corner of the menu items by adding a span element in the `contextMenuBeforeItemRender` event.

INDEX.JS

```

var nodes = [{
    id: 'node1',
    width: 100,
    height: 100,
    offsetX: 100,
    offsetY: 100,
    style: {
        fill: '#6BA5D7',
        strokeColor: 'white',
        strokeWidth: 1
    },
    annotations: [{

```



```

        id: 'label1',
        content: 'Rectangle1',
        offset: {
            x: 0.5,
            y: 0.5
        },
        style: {
            color: 'white'
        }
    }
}
},
{
    id: 'node2',
    width: 100,
    height: 100,
    offsetX: 300,
    offsetY: 100,
    style: {
        fill: '#6BA5D7',
        strokeColor: 'white',
        strokeWidth: 1
    },
    annotations: [{
        id: 'label2',
        content: 'Rectangle2',
        offset: {
            x: 0.5,
            y: 0.5
        },
        style: {
            color: 'white'
        }
    }]
},
];
var connector = {
    id: 'connector1',
    sourceID: 'node1',
    targetID: 'node2',
    type: 'Straight',
    style: {
        strokeColor: '#6BA5D7',
        fill: '#6BA5D7',
        strokeWidth: 2,
        targetDecorator: {
            style: {
                fill: '#6BA5D7',
                strokeColor: '#6BA5D7'
            }
        }
    }
};
var diagram = new ej.diagrams.Diagram({
    width: '100%',
    height: '350px',
    nodes: nodes,
    connectors: [connector],

```

```

contextMenuSettings: {
    //Enables the context menu
    show: true,
    items: [{
        text: 'delete',
        id: 'delete'
    }],
    // Hides the default context menu items
    showCustomMenuOnly: false,
},
contextMenuOpen: function(args) {
    //do your custom action here.
    for (let item of args.items) {
        if (item.text === 'delete') {
            if (!diagram.selectedItems.nodes.length &&
!diagram.selectedItems.connectors.length) {
                args.hiddenItems.push(item.text);
            }
        }
    }
},
contextMenuClick: function(args) {
    //do your custom action here.
    if (args.item.id === 'delete') {
        if ((diagram.selectedItems.nodes.length +
diagram.selectedItems.connectors.length) > 0) {
            diagram.cut();
        }
    }
},
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

```

```
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Context menu events

You would be notified with events, when you try to open the context menu items [contextMenuOpen](#) and when you click the menu items `contextMenuClick`. The following code example illustrates how to define those events.

INDEX.JS

```
var nodes = [{
    id: 'node1',
    width: 100,
    height: 100,
    offsetX: 100,
    offsetY: 100,
    style: {
        fill: '#6BA5D7',
        strokeColor: 'white',
        strokeWidth: 1
    },
    annotations: [{
        id: 'label1',
        content: 'Rectangle1',
        offset: {
            x: 0.5,
            y: 0.5
        },
        style: {
            color: 'white'
        }
    }]
},
{
    id: 'node2',
    width: 100,
    height: 100,
    offsetX: 300,
    offsetY: 100,
    style: {
```

```

        fill: '#6BA5D7',
        strokeColor: 'white',
        strokeWidth: 1
    },
    annotations: [{
        id: 'label2',
        content: 'Rectangle2',
        offset: {
            x: 0.5,
            y: 0.5
        },
        style: {
            color: 'white'
        }
    }]
},
];
var connector = {
    id: 'connector1',
    sourceID: 'node1',
    targetID: 'node2',
    type: 'Straight',
    style: {
        strokeColor: '#6BA5D7',
        fill: '#6BA5D7',
        strokeWidth: 2,
        targetDecorator: {
            style: {
                fill: '#6BA5D7',
                strokeColor: '#6BA5D7'
            }
        }
    }
};
var diagram = new ej.diagrams.Diagram({
    width: '100%',
    height: '350px',
    nodes: nodes,
    connectors: [connector],
    contextMenuSettings: {
        //Enables the context menu
        show: true,
        items: [{
            text: 'delete',
            id: 'delete'
        }],
        // Hides the default context menu items
        showCustomMenuOnly: false,
    },
    contextMenuOpen: function(args) {
        //do your custom action here.
        for (let item of args.items) {
            if (item.text === 'delete') {
                if (!diagram.selectedItems.nodes.length &&
!diagram.selectedItems.connectors.length) {
                    args.hiddenItems.push(item.text);
                }
            }
        }
    }
});

```

```

    }
  },
  contextMenuClick: function(args) {
    //do your custom action here.
    if (args.item.id === 'delete') {
      if ((diagram.selectedItems.nodes.length +
diagram.selectedItems.connectors.length) > 0) {
        diagram.cut();
      }
    }
  },
  '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Overview in ##Platform_Name## Diagram control

Overview control allows you to see a preview or an overall view of the entire content of a diagram. This helps you to look at the overall picture of a large diagram and also to navigate, pan, or zoom, on a particular position of the page.

When you work on a very large diagram, you may not know the part you are actually working on, or navigation from one part to another might be difficult. One solution for navigation is to zoom out the entire diagram and find where you are. Then, you can zoom in a particular area you want to. This solution is not suitable when you need some frequent navigation.

Overview control solves these problems by showing a preview, that is, an overall view of the entire diagram. A rectangle indicates viewport of the diagram. Navigation becomes easy by dragging this rectangle.

Create overview

The `sourceID` property of overview should be set with the corresponding diagram ID for the overall view.

The `width` and `height` properties of the overview allow you to define the size of the overview.

The following code illustrates how to create overview.

Zoom and Pan

In overview, the view port of the diagram is highlighted with a red colored rectangle. Diagram can be zoomed/panned by interacting with that. You can interact with overview as follows:

- Resize the rectangle: Zooms in/out the diagram.
- Drag the rectangle: Pans the diagram.
- Click at a position: Navigates to the clicked region.
- Choose a particular region by clicking and dragging: Navigates to the specified region.

The following image shows how the diagram is zoomed/panned with overview.

INDEX.JS

```
ej.diagrams.Diagram.Inject(ej.diagrams.DataBinding,
ej.diagrams.HierarchicalTree);
// A node is created and stored in nodes array.
var data = [
  {
    'Id': 'parent', 'Name': 'Maria Anders', 'Designation': 'Managing
Director',
    'ImageUrl': '../content/images/orgchart/Clayton.png', 'IsExpand':
'true', 'RatingColor': '#C34444'
  },
  {
    'Id': 1, 'Name': 'Ana Trujillo', 'Designation': 'Project Manager',
    'ImageUrl': '../content/images/orgchart/Thomas.PNG', 'IsExpand':
'false',
    'RatingColor': '#68C2DE', 'ReportingPerson': 'parent'
  },
  {
    'Id': 2, 'Name': 'Anto Moreno', 'Designation': 'Project Lead',
    'ImageUrl': '../content/images/orgchart/image53.png', 'IsExpand':
'false',
```

```

        'RatingColor': '#93B85A', 'ReportingPerson': 'parent'
    },
    {
        'Id': 3, 'Name': 'Thomas Hardy', 'Designation': 'Senior S/w Engg',
        'ImageUrl': '../content/images/orgchart/image57.png', 'IsExpand':
    'false',
        'RatingColor': '#68C2DE', 'ReportingPerson': 1
    },
    {
        'Id': 4, 'Name': 'Christina kaff', 'Designation': 'S/w Engg',
        'ImageUrl': '../content/images/orgchart/Robin.png', 'IsExpand':
    'false',
        'RatingColor': '#93B85A', 'ReportingPerson': 3
    }
  ]
}
var items = new ej.data.DataManager(data , new ej.data.Query().take(7));
var overview;
var diagram = new ej.diagrams.Diagram({
  snapSettings: { constraints: 0 },
  layout: {
    type: 'OrganizationalChart', margin: { top: 20 },
    getLayoutInfo: (node, tree) => {
      if (!tree.hasSubTree) {
        tree.orientation = 'Vertical';
        tree.type = 'Alternate';
      }
    }
  },
  dataSourceSettings: {
    id: 'Id', parentId: 'ReportingPerson', dataManager: items
  },
  getNodeDefaults: (obj) => {
    obj.height = 50;
    obj.backgroundColor = 'lightgrey';
    obj.style = { fill: 'transparent', strokeWidth: 2 };
    return obj;
  },
  getConnectorDefaults: (connector) => {
    connector.targetDecorator.shape = 'None';
    connector.type = 'Orthogonal';
    return connector;
  },
  setNodeTemplate: (obj) => {
    var content = new ej.diagrams.StackPanel();
    content.id = obj.id + '_outerstack';
    content.style.strokeColor = 'darkgreen';
    content.orientation = 'Horizontal';
    content.padding = { left: 5, right: 10, top: 5, bottom: 5 };
    var image = new ej.diagrams.ImageElement();
    image.width = 50;
    image.height = 50;
    image.style.strokeColor = 'none';
    image.source = './employee.PNG';
    image.id = obj.id + '_pic';
    var innerStack = new ej.diagrams.StackPanel();
    innerStack.style.strokeColor = 'none';
    innerStack.margin = { left: 5, right: 0, top: 0, bottom: 0 };
    innerStack.id = obj.id + '_innerstack';
    var text = new ej.diagrams.TextElement();

```

```

        text.content = obj.data['Name'];
        text.style.color = 'blue';
        text.style.strokeColor = 'none';
        text.style.fill = 'none';
        text.id = obj.id + '_text1';
        var desigText = new ej.diagrams.TextElement();
        desigText.margin = { left: 0, right: 0, top: 5, bottom: 0 };
        desigText.content = obj.data['Designation'];
        desigText.style.color = 'blue';
        desigText.style.strokeColor = 'none';
        desigText.style.fill = 'none';
        desigText.style.textWrapping = 'Wrap';
        desigText.id = obj.id + '_desig';
        innerStack.children = [text, desigText];
        content.children = [image, innerStack];
        return content;
    }
});
diagram.appendTo('#element');
overview = new ej.diagrams.Overview({
    sourceID: 'element'
});
overview.appendTo('#overview');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Diagram</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
diagrams/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div style="width:74%;height: 500px; float:left">
```



```

        <div id="element"></div>
    </div>
    <div style="width:25%;height:200px;float:left; border-
color:lightgray;border-style:solid;">
        <div id="overview"></div>

</div></div><script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Dialog

Template in ##Platform_Name## Dialog control

In Dialog the template support is provided to the header and footer sections. So any text or HTML content can be appending in these sections.

Header

The Dialog header content can be provided through the [header](#) property, and it will allow both text and any HTML content as a string. Also in header, close button is provided as built-in support, and this can be enabled through the [showCloseIcon](#) property.

Footer

The Dialog footer can be enabled by adding built-in [buttons](#) or providing any HTML string through the [footerTemplate](#).

The [buttons](#) and [footerTemplate](#) properties can't be used at the same time.

Content

The Dialog content can be provided through the [content](#) property, and it accepts both text and HTML string as content.

The below example demonstrates the usage of header, footer and content templates in the Dialog control.

INDEX.JS

```

/**
 * Dialog template sample
 */
var icontemp = '<button id="sendButton" class="e-control e-btn e-primary"
data-ripple="true">' + 'Send</button>';
var headerimg = '';
var sendbutton = new ej.buttons.Button();
var proxy = this;
// Initialize Dialog component
var dialog = new ej.popups.Dialog({
    // Enables the header with template content

```

```

    header: headerimg + '<div class="dlg-template" title="Nancy" class="e-
icon-settings"> Nancy </div>',
    // Enables the footer with template content
    footerTemplate: ' <input id="inVal" class="e-input" type="text"
placeholder="Enter your message here!"/>' + icontemp,
    // Dialog content
    content: document.getElementById("dlgContent"),
    // Enables the close icon button in header
    showCloseIcon: true,
    // The Dialog shows within the target element
    target: document.getElementById("container"),
    //Dialog width
    width: '400px',
    height: '250px',
    beforeOpen: onBeforeopen
});
// Render initialized Dialog
dialog.appendTo('#dialog');
sendbutton.appendTo('#sendButton');
// Sample level code to handle the button click action
document.getElementById('targetButton').onclick = function () {
    // Call the show method to open the Dialog
    dialog.show();
};
document.getElementById('sendButton').onkeydown = function (e) {
    if (e.keyCode === 13) {
        updateTextValue();
    }
};
document.getElementById('inVal').onkeydown = function (e) {
    if (e.keyCode === 13) {
        updateTextValue();
    }
};
document.getElementById('sendButton').onclick = function () {
    updateTextValue();
};
function onBeforeopen() {
    document.getElementById('dlgContent').style.visibility = 'visible';
}
function updateTextValue() {
    var enteredVal = document.getElementById('inVal');
    var dialogTextElement =
document.getElementsByClassName('dialogText')[0];
    var dialogTextWrap =
document.getElementsByClassName('dialogContent')[0];
    if (enteredVal.value !== '') {
        dialogTextElement.innerHTML = enteredVal.value;
        enteredVal.value = '';
    }
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
<title>Essential JS 2 Dialog with template support</title>

```

```

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="TypeScript UI Components">
<meta name="author" content="Syncfusion">
<link href="styles.css" rel="stylesheet">
<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button class="e-control e-btn" id="targetButton" role="button" e-
ripple="true">Open Dialog</button>
        <div id="dialog" class="custom-template"></div>
        <div id="dlgContent" style="visibility: hidden"
class="dialogContent">
            <span class="dialogText">
                Greetings Nancy! When will you share me the source files of the
project?
            </span>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [How to add an icon to dialog buttons](#)
- [How to customize the dialog appearance](#)

Animation in ##Platform_Name## Dialog control

The Dialog can be animated during the open and close actions. Also, user can customize animation's [delay](#), [duration](#) and [effect](#) by using [animationSettings](#) property.

<!-- markdownlint-disable MD033 -->

delay	The Dialog animation will start with the mentioned delay
duration	Specifies the animation duration to complete with one animation cycle
effect	<p>Specifies the animation effects of Dialog open and close actions effect.</p> <p>List of supported animation effects: 'Fade' 'FadeZoom' 'FlipLeftDown' 'FlipLeftUp' 'FlipRightDown' 'FlipRightUp' 'FlipXDown' 'FlipXUp' 'FlipYLeft' 'FlipYRight' 'SlideBottom' 'SlideLeft' 'SlideRight' 'SlideTop' 'Zoom' 'None'</p> <p>If the user sets 'Fade' effect, then the Dialog will open with 'FadeIn' effect and close with 'FadeOut' effect</p>

In the below sample, **Zoom** effect is enabled. So, The Dialog will open with **ZoomIn** and close with **ZoomOut** effects.

INDEX.JS

```
ej.base.enableRipple(true);
// Initialize Dialog component
var dialog = new ej.popups.Dialog({
  //Animation options
  animationSettings: {
    effect: 'Zoom',
    duration: 400,
    delay: 0
  },
  // Enables the header
  header: 'Dialog',
  // Dialog content
  content: 'Dialog enabled with Zoom effect',
  // Enables the footer buttons
  buttons: [
    {
      // Click the footer buttons to hide the Dialog
      'click': () => { dialog.hide(); },
      // Accessing button component properties by buttonModel property
      buttonModel: {
        content: 'OK',
        isPrimary: true
      }
    },
    {
      'click': () => { dialog.hide(); },
      buttonModel: {
        content: 'Cancel'
      }
    }
  ],
  // The Dialog shows within the target element
  target: document.getElementById("container"),
```

```

    // Dialog width
    width: '250px'
  });
  // Render initialized Dialog
  dialog.appendTo('#dialog');
  // Sample level code to handle the button click action
  document.getElementById('targetButton').onclick = function() {
    // Call the show method to open the Dialog
    dialog.show();
  }

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Dialog Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript UI Components">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <button class="e-control e-btn" id="targetButton" role="button" e-
ripple="true" style="position: absolute;">Open Dialog</button>
    <div id="dialog"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Resize in ##Platform_Name## Dialog control

The Dialog supports resizing feature. To resize the dialog, we have to select and resize it by using its handle (grip) or hovering on any of the edges or borders of the dialog within the sample container.

The resizable dialog can be created by setting the [enableResize](#) property to true, which is used to change the size of a dialog dynamically and view its content with expanded mode. The [resizeHandles](#) property can also be configured for all the which directions in which the dialog should be resized. When you configure the target property along with the [enableResize](#) property, the dialog can be resized within its specified target container.

INDEX.JS

```
ej.base.enableRipple(true);
var dialog = new ej.popups.Dialog({
  // Enables the draggable option
  allowDragging: true,
  // Enables the resize option
  enableResize: true,
  // Enables resize in all the direction
  resizeHandles: ['All'],
  // Enables the header
  header: 'Dialog',
  // Dialog content
  content: 'This is a Dialog with resize enabled',
  // Enables the draggable option
  allowDragging: true,
  // Enables the footer buttons,
  buttons: [
    {
      // Click the footer buttons to hide the Dialog
      'click': () => {
        dialog.hide();
      },
      // Accessing button component properties by buttonModel property
      buttonModel: {
        // Enables the primary button
        isPrimary: true,
        content: 'OK'
      }
    },
    {
      'click': () => {
        dialog.hide();
      },
      buttonModel: {
        content: 'Cancel'
      }
    }
  ],
  // The Dialog shows within the target element
  target: document.getElementById("container"),
  // Dialog width
  width: '250px',
});
// Render initialized Dialog
dialog.appendTo('#dialog');
```

```
// Sample level code to handle the button click action
document.getElementById('targetButton').onclick = function() {
    // Call the show method to open the Dialog
    dialog.show();
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Dialog with resize support</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript UI Components">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="padding: 20px;">
    <button class="e-control e-btn" id="targetButton" role="button" e-
ripple="true" style="position: absolute;">Open Dialog</button>
    <div id="dialog"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Dialog utility in ##Platform_Name## Dialog control

The dialog component provides built-in utility functions to render the alert and confirm dialogs with the minimal code. The following options are used as an argument on calling the utility functions:

Options	Description
---------	-------------

|-----|-----|

| title | Specifies the title of dialog like the [header](#) property. |

| content | Specifies the value that can be displayed in dialog's content area like the [content](#) property. |

| isModal | Specifies the Boolean value whether the dialog can be displayed as modal or non-modal. For more details, refer to the [isModal](#) property. |

| position | Specifies the value where the alert or confirm dialog is positioned within the document. For more details, refer to the [position](#) property { X: 'center', Y: 'center' } |

| okButton | Configures the OK button that contains button properties with the click events.
 okButton:{ icon:'prefix icon to the button', cssClass:'custom class to the button', click: 'action for OK button click', text: 'Yes' // <-- Default value is 'OK' } |

| cancelButton | Configures the Cancel button that contains button properties with the click events.
 cancelButton:{ icon:'prefix icon to the button', cssClass:'custom class to the button', click: 'action for 'Cancel' button click', text: 'No' // <-- Default value is 'Cancel' } |

| isDraggable | Specifies the value whether the alert or confirm dialog can be dragged by the user. |

| showCloseIcon | When set to true, the close icon is shown in the dialog component. |

| closeOnEscape | When set to true, you can close the dialog by pressing ESC key. |

| animationSettings | Specifies the animation settings of the dialog component. |

| cssClass | Specifies the CSS class name that can be appended to the dialog. |

| zIndex | Specifies the order of the dialog, that is displayed in front or behind of another component. |

| open | Event which is triggered after the dialog is opened. |

| Close | Event which is triggered after the dialog is closed. |

Alert dialog

An alert dialog box is used to display warning like messages to the users. Use the following code to render a simple alert dialog in an application.

INDEX.JS

```
ej.base.enableRipple(true);
document.getElementById('targetButton').onclick = function(){
    // Initialize and render alert dialog
    ej.popups.DialogUtility.alert('This is an Alert Dialog!');
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Dialog utility</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript UI Components">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
```



```

<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <button class="e-control e-btn" id="targetButton" role="button" e-
ripple="true">Open Alert Dialog</button>
    <div id="dialog"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Render an alert dialog with options

INDEX.JS

```

ej.base.enableRipple(true);
document.getElementById('targetButton').onclick = function(){
  // Initialize and render alert dialog with options
  ej.popups.DialogUtility.alert({
    title: 'Alert Dialog',
    content: "This is an Alert Dialog!",
    okButton: { text: 'OK', click: okClick },
    showCloseIcon: true,
    closeOnEscape: true,
    animationSettings: { effect: 'Zoom' }
  });
};
function okClick(){
  alert('you clicked OK button');
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Dialog utility</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript UI Components">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <button class="e-control e-btn" id="targetButton" role="button" e-
ripple="true">Open Alert Dialog</button>
    <div id="dialog"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Confirm dialog

A confirm dialog displays a specified message along with 'OK' and 'Cancel' button.

INDEX.JS

```

ej.base.enableRipple(true);
document.getElementById('targetButton').onclick = function(){
  // Initialize and render confirm dialog
  ej.popups.DialogUtility.confirm('This is a Confirmation Dialog!');
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Dialog utility</title>

```

```

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="TypeScript UI Components">
<meta name="author" content="Syncfusion">
<link href="styles.css" rel="stylesheet">
<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button class="e-control e-btn" id="targetButton" role="button" e-
ripple="true">Open Confirm Dialog</button>
        <div id="dialog"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Render a confirmation dialog with options

INDEX.JS

```

ej.base.enableRipple(true);
document.getElementById('targetButton').onclick = function(){
// Initialize and render confirm dialog with custom options
ej.popups.DialogUtility.confirm({
    title: ' Confirmation Dialog',
    content: "This is a Confirmation Dialog!",
    okButton: { text: 'OK', click: okClick },
    cancelButton: { text: 'Cancel', click: cancelClick },
    showCloseIcon: true,
    closeOnEscape: true,
    animationSettings: { effect: 'Zoom' }
});
};
function okClick() {

```

```

    alert('you clicked OK button');
    //Hide the dialog
    this.hide();
}
function cancelClick(){
    alert('you clicked Cancel button');
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Dialog utility</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript UI Components">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <button class="e-control e-btn" id="targetButton" role="button" e-
ripple="true">Open Confirm Dialog</button>
    <div id="dialog"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Close utility dialog

When rendering an Alert and Confirmation dialog through utility methods, You can close the dialog using the following ways.

- By pressing the escape key if the "closeOnEscape" property is enabled.
- By clicking the close button if the "showCloseIcon" property is enabled.

You can also manually close the Dialogs by creating an instance to the dialog and call the "hide" method.

Below sample demonstrates the different ways of hiding the utility dialog.

INDEX.JS

```
ej.base.enableRipple(true);
document.getElementById('targetButton').onclick = function(){
  // Initialize and render confirm dialog with custom options
  ej.popups.DialogUtility.confirm({
    title: ' Confirmation Dialog',
    content: "This is a Confirmation Dialog!",
    okButton: { text: 'OK', click: okClick },
    cancelButton: { text: 'Cancel', click: cancelClick },
    showCloseIcon: true,
    closeOnEscape: true,
    animationSettings: { effect: 'Zoom' }
  });
};
function okClick(){
  alert('you clicked OK button');
  //Hide the dialog
  this.hide();
}
function cancelClick(){
  alert('you clicked Cancel button');
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Dialog utility</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript UI Components">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button class="e-control e-btn" id="targetButton" role="button" e-
ripple="true">Open Confirm Dialog</button>
        <div id="dialog"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Style in ##Platform_Name## Dialog control

The following content provides the exact CSS structure that can be used to modify the control's appearance based on the user preference.

Customizing the dialog header

Use the following CSS to customize the dialog header properties.

```

`css
.e-dialog .e-dlg-header {
color: green;
font-size: 20px;
font-weight: normal;
}
`

```

Customizing the dialog content

Use the following CSS to customize the dialog content properties.

```

`css
.e-dialog .e-dlg-content {
color: red;
font-size: 10px;
font-weight: normal;
line-height: normal;
}
`

```

Customizing modal dialog overlay

Use the following CSS to customize the modal dialog overlay.

```
`css
.e-dlg-overlay {
background-color: slategray;
opacity: 0.6;
}
`
```

Customizing the dialog resize icon

Use the following CSS to customize the dialog resize icon.

```
`css
/ To change the icon content /
.e-dialog .e-south-east::before, .e-dialog .e-south-west::before {
content: '\f047';
}
/ To set the icon pack /
.e-dialog .e-resize-handle {
font: normal normal normal 14px/1 FontAwesome;
}
`
```

The above CSS demonstration uses the font awesome icon.

Customizing the dialog close button

Use the following CSS to customize the dialog close button.

```
`css
/ To specify font size and color /
.e-dialog .e-btn .e-btn-icon.e-icon-dlg-close {
font-size: 12px;
color: red;
}
`
```

Customizing the dialog footer button

Use the following CSS to customize the dialog footer button.

```
`css
/ To specify font color, background color and border color /
```

```
.e-btn.e-flat.e-primary, .e-css.e-btn.e-flat.e-primary {
background-color: transparent;
border-color: transparent;
color: blue;
}
,
```

Localization in ##Platform_Name## Dialog control

Localization library allows to localize the default text content of Dialog. In Dialog, The close button's tooltip text alone will be localize based on culture. By using [locale](#) property you can the culture dynamically in dialog component.

| Locale key | en-US (default) |

|-----|-----|

| close | Close |

Loading translations

To load translation object in an application use **load** function of **L10n** class.

In the below sample, **French** culture is set to Dialog and change the close button's tooltip text.

INDEX.JS

```
ej.base.enableRipple(true);
// Load French culture for Dialog close button tooltip text
ej.base.L10n.load({
  'fr-BE': {
    'dialog': {
      'close': "Fermer"
    }
  }
});
// Initialization of Dialog component
var dialog = new ej.popups.Dialog({
  // Set the locale culture
  locale: 'fr-BE',
  // Enables the header
  header: 'Dialogue',
  // Enables the close icon button in header
  showCloseIcon: true,
  // Dialog content
  content: 'Dialogue avec la culture française',
  // The Dialog shows within the target element
  target: document.getElementById("container"),
  // Dialog width
  width: '250px',
});
// Render initialized Dialog
dialog.appendTo('#dialog');
// Sample level code to handle the button click action
document.getElementById('targetButton').onclick = function() {
  // Call the show method to open the Dialog
```



```

    dialog.show();
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Dialog with locale support</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript UI Components">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <button class="e-control e-btn" id="targetButton" role="button" e-
ripple="true" style="position: absolute;">Open Dialog</button>
    <div id="dialog"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Accessibility in ##Platform_Name## Dialog control

The Dialog component followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Dialog component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2 Support](#) | |

| [Section 508 Support](#) | |

| [Screen Reader Support](#) | |

| [Right-To-Left Support](#) | |

| [Color Contrast](#) | |

| [Mobile Device Support](#) | |

| [Keyboard Navigation Support](#) | |

| [Accessibility Checker Validation](#) | |

| [Axe-core Accessibility Validation](#) | |

```
<style>
.post .post-content img {
display: inline-block;
margin: 0.5em 0;
}
</style>
<div> - All
features of the component meet the requirement.</div>
<div> - Some features of the component do not meet the requirement.</div>
<div> - The
component does not meet the requirement.</div>
```

WAI-ARIA attributes

The Dialog characterized with complete ARIA Accessibility support which helps to accessible by on-screen readers and other assistive technology devices. This component designed with the reference of the guidelines document given in [WAI ARAI Accessibility Practices](#).

The Dialog component uses the **Dialog** role and following ARIA properties to its element based on its state.

| Property | Functionalities |

| --- | --- |

| aria-describedby | It indicates the Dialog content description is notified to the user through assistive technologies. |

| aria-labelledby | The Dialog title is notified to the user through assistive technologies. |

| aria-modal | For modal dialog it's value is true and non-modal dialog its value is false |

| aria-grabbed | Enable the draggable Dialog and starts dragging it is value is true and stopping the drag its value is false |

Keyboard interaction

Keyboard interaction of Dialog component has designed based on [WAI-ARIA Practices](#) described for Dialog. User can use the following shortcut keys to interact with the Dialog.

<!-- markdownlint-disable MD033 -->

Keyboard shortcuts	Actions
Esc	Closes the Dialog. This functionality can be controlled by using closeOnEscape
Enter	When the Dialog button or any input (except text area) is in focus state, when pressing the Enter key, the click event associated with the primary button or button will trigger. Enter key is not working when the Dialog content contains any text area with initial focus
Ctrl + Enter	When the Dialog content contains text area and it is in focus state, and press the Ctrl + Enter key to call the click event function associated with primary button
Tab	Focus will be changed within the Dialog elements
Shift + Tab	The Focus will be changed previous focusable element within the Dialog elements. When focusing a first focusable element in the Dialog, then press the shift + tab key, it will change the focus to last focusable element

INDEX.JS

```
ej.base.enableRipple(true);
// Initialize Dialog component
var dialog = new ej.popups.Dialog({
  // Enables the header
  header: 'Feedback',
  // Dialog content
  content: document.getElementById("dlgContent"),
  // Enables the close icon in header
  showCloseIcon: true,
  // Enables the footer buttons
  buttons: [{
    // Accessing button component properties by buttonModel property
    buttonModel: {
      // Enables the primary button
      isPrimary: true,
      content: 'Submit'
    },
    // Click the footer buttons to hide the Dialog
    click: function () {
```

```

        this.hide();
    }
    }],
    // The Dialog shows within the target element
    target: document.getElementById("container"),
    // Dialog width
    width: '400px',
    // Dialog height
    height: '330px',
    beforeOpen: onBeforeopen
});
// Render initialized Dialog
dialog.appendTo('#dialog');
// Sample level code to handle the button click action
document.getElementById('targetButton').onclick = function () {
    // Call the show method to open the Dialog
    dialog.show();
};
function onBeforeopen() {
    document.getElementById('dlgContent').style.visibility = 'visible';
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Dialog</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript UI Components">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <button class="e-control e-btn" id="targetButton" role="button" e-
ripple="true" style="position: absolute;">Open Dialog</button>
    <div id="dialog"></div>
    <form id="dlgContent" style="visibility: hidden">

```

```

        <div class="form-group"><label for="email">Email:</label>
        <input type="email" class="form-control" id="email">
        </div>
        <div class="form-group">
            <label for="comment">Comments:</label>
            <textarea style="resize: none;" class="form-control"
rows="4" id="comment"></textarea>
        </div>
    </form>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Show dialog with fullscreen](#)

Ensuring accessibility

The Dialog component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Dialog component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Dialog component with accessibility tools.

See also

- [Accessibility in Syncfusion components](#)

How To

Create nested dialog in ##Platform_Name## Dialog control

A Dialog can be nested within another Dialog. The below sample contains parent and child Dialog (inner Dialog).

Step 1:

Create two div elements with id `#dialog` and `#innerDialog`.

Step 2:

Initialize the Dialog as mentioned in the below sample.

Step 3:

Set the inner Dialog target as `#dialog`.

INDEX.JS

```

ej.base.enableRipple(true);
// Initialize the Outer Dialog component

```

```

var dialog = new ej.popups.Dialog({
    // Enables the header
    header: 'Outer Dialog',
    // Enables the close icon button in header
    showCloseIcon: true,
    // The Dialog shows within the target element
    target: document.getElementById("container"),
    // Dialog content
    content: document.getElementById("dlgContent"),
    //Dialog height
    height: '300px',
    animationSettings: { effect: 'None' },
    // Disable the Esc key option to hide the Dialog
    closeOnEscape: false,
    //Dialog width
    width: '400px',
    // Dialog beforeOpen event
    beforeOpen: onBeforeopen
});
// Render initialized outer Dialog
dialog.appendTo('#dialog');
// Sample level code to handle the button click action
document.getElementById('targetButton').onclick = function () {
    // Call the show method to open the Dialog
    dialog.show();
};
// initialize the Inner Dialog component
var innerDialog = new ej.popups.Dialog({
    // Enables the header
    header: 'Inner Dialog',
    // Enables the close icon button in header
    showCloseIcon: true,
    animationSettings: { effect: 'None' },
    // Disable the Esc key option to hide the Dialog
    closeOnEscape: false,
    // Dialog content
    content: 'This is a Nested Dialog',
    // InnerDialog target as outerDialog
    target: document.getElementById("dialog"),
    // Dialog height
    height: '150px',
    // Dialog Width
    width: '250px'
});
document.getElementById('innerButton').onclick = function () {
    // Call the show method to open the Dialog
    innerDialog.show();
};
// Render initialized inner Dialog
innerDialog.appendTo('#innerDialog');
function onBeforeopen() {
    document.getElementById('dlgContent').style.visibility = 'visible';
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Nested Dialog</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript UI Components">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <button class="e-control e-btn" id="targetButton" role="button">Open
Dialog</button>
    <div id="dialog">
      </div>
      <div id="innerDialog"></div>
      <div id="dlgContent" style="visibility: hidden">
        <button class="e-control e-btn" id="innerButton"
role="button">Open InnerDialog</button>
      </div>
    </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Position the dialog on center of the page on scrolling in `##Platform_Name##` Dialog control

By default, when scroll the page/container Dialog also scrolled along with the page/container. When a user expects to display the Dialog in the same position without scrolling achieving this in sample level as like below. Here added 'e-fixed' class to Dialog element by using [cssClass](#) property and prevent the scrolling.

INDEX.JS

```

ej.base.enableRipple(true);
// Initialize Dialog component
var dialog = new ej.popups.Dialog({
  // Enables the header
  header: 'Dialog',
  // Dialog content

```

```

        content: document.getElementById("dlgContent"),
        // Disable the Esc key to hide the Dialog
        closeOnEscape: false,
        // The Dialog shows within the target element
        target: document.getElementById("container"),
        // Dialog width
        width: '250px',
        beforeOpen: onBeforeopen
    });
    // Render initialized Dialog
    dialog.appendTo('#dialog');
    // Sample level code to prevent Dialog scrolling
    document.getElementById('targetButton').onclick = function() {
        dialog.cssClass = 'e-fixed';
    }
    function onBeforeopen() {
        document.getElementById('dlgContent').style.visibility = 'visible';
    }

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Dialog with scrollable content</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="TypeScript UI Components">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="styles.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <b>JavaScript:</b><br>
            JavaScript is a high-level, dynamic, untyped, and interpreted
programming language. It has been standardized in the ECMAScript
            language specification. Alongside HTML and CSS, it is one of the
three essential technologies of World Wide Web
            content production; the majority of websites employ it and it is
supported by all modern Web browsers without
            plug-ins. JavaScript is prototype-based with first-class
functions, making it a multi-paradigm language, supporting
            object - oriented , imperative, and functional programming
styles.

```



```

        <br><br><br>
        <b>MVC:</b><br>
        Model-view-controller (MVC) is a software architecture pattern
        which separates the representation of information from the user's
        interaction with it. The model consists of application data, business rules,
        logic, and functions. A view can be any output representation of data, such
        as a chart or a diagram. Multiple views of the same data are possible, such
        as a bar chart for management and a tabular view for accountants. The
        controller mediates input, converting it to commands for the model or
        view. The central ideas behind MVC are code reusability and in addition to
        dividing the application into three kinds of components, the MVC design
        defines the interactions between them.

        A controller can send commands to its associated view to change
        the view's presentation of the model (e.g., by scrolling through a
        document). It can also send commands to the model to update the model's
        state (e.g., editing a document).

        A model notifies its associated views and controllers when there
        has been a change in its state. This notification allows the views to
        produce updated output, and the controllers to change the available set of
        commands. A passive implementation of MVC omits these notifications, because
        the application does not require them or the software platform does not
        support them.

        A view requests from the model the information that it needs to
        generate an output representation to the user.
    </div>
    <div id="dialog"></div>
    <div id="dlgContent" style="visibility: hidden">
        <button class="e-control e-btn" id="targetButton"
        role="button">Prevent Dialog Scroll</button>
    </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Load dialog content using ajax in ##Platform_Name## Dialog control

You can load dialog's content dynamically from external source like external file using AJAX library. The AJAX library can make the request and load dialog's content using its **success** event. Refer the following link to learn about how to load dialog content using AJAX.

[AJAX Content](#)

Render a dialog without header in ##Platform_Name## Dialog control

The dialog can be rendered without header by setting the [header](#) property value as empty string or null. By default, dialog is rendered without header.

INDEX.JS

```

ej.base.enableRipple(true);
// Initialization of Dialog component
var dialog = new ej.popups.Dialog({

```

```

        buttons: [
            {
                // Click the footer buttons to hide the Dialog
                'click': function () {
                    dialog.hide();
                },
                // Accessing button component properties by buttonModel property
                buttonModel: {
                    // Enables the primary button
                    isPrimary: true,
                    content: 'OK'
                }
            },
            {
                'click': function () {
                    dialog.hide();
                },
                buttonModel: {
                    content: 'Cancel',
                    cssClass: 'e-flat'
                }
            }
        ],
        // Dialog content
        content: 'This is a dialog without header',
        // The Dialog shows within the target element
        target: document.getElementById("container"),
        // Dialog width
        width: '250px',
    });
    // Sample level code to handle the button click action
    document.getElementById('targetButton').onclick = function () {
        // Call the show method to open the Dialog
        dialog.show();
    };
    // Render initialized Dialog
    dialog.appendTo('#dialog');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Dialog with header</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="TypeScript UI Components">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button class="e-control e-btn" id="targetButton" role="button" e-
ripple="true" style="position: absolute;">Open Dialog</button>
        <div id="dialog"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Show dialog with full screen in ##Platform_Name## Dialog control

You can show the dialog in fullscreen by passing `true` as argument to the dialog `show` method. By using `visible` property you can prevent the dialog from initially shown.

INDEX.JS

```

ej.base.enableRipple(true);
// Initialize Dialog component
var dialog = new ej.popups.Dialog({
    // Enables the footer buttons
    buttons: [
        {
            // Click the footer buttons to hide the Dialog
            'click': function () {
                dialog.hide();
            },
            // Accessing button component properties by buttonModel property
            buttonModel: {
                //Enables the primary button
                isPrimary: true,
                cssClass: 'e-flat',
                content: 'OK'
            }
        },
        {
            'click': function () {
                dialog.hide();
            },
            buttonModel: {
                content: 'Cancel',
                cssClass: 'e-flat'
            }
        }
    ]
});

```

```

    }
  },
  // Enables the header
  header: 'Dialog',
  // Dialog content
  content: 'This is a Dialog with fullscreen display',
  // The Dialog shows within the target element
  target: document.getElementById("container"),
  // Dialog width
  width: '250px',
  visible: false,
});
// Render initialized Dialog
dialog.appendTo('#dialog');
// Sample level code to handle the button click action
document.getElementById('targetButton').onclick = function () {
  // Call the show method to open the Dialog
  dialog.show(true);
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Dialog with fullscreen</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript UI Components">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <button class="e-control e-btn" id="targetButton" role="button" e-
ripple="true" style="position: absolute;">Open Dialog</button>
    <div id="dialog"></div>
  </div>
</body>
</html>

```

```

var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Display a dialog with custom position in ##Platform_Name## Dialog control

By default, the dialog is displayed in the center of the target container. The dialog position can be set using the [position](#) property by providing custom X and Y coordinates. The dialog can be positioned inside the target based on the given X and Y values.

INDEX.JS

```

ej.base.enableRipple(true);
var firstDialog = new ej.popups.Dialog ({
    // Enables the header
    header: 'Position-01',
    // The Dialog shows within the target element
    target: document.getElementById("container"),
    // Dialog content
    content: 'The dialog is positioned at {X: 420, Y: 14} coordinates',
    //Dialog height
    height: '120px',
    //Dialog width
    width: '360px',
    position: { X: 420, Y: 14 },
    animationSettings: { effect: 'None' }
});
// Render initialized outer Dialog
firstDialog.appendTo('#firstDialog');
var secondDialog = new ej.popups.Dialog ({
    // Enables the header
    header: 'Position-02',
    animationSettings: { effect: 'None' },
    // Dialog content
    content: 'The dialog is positioned at {X: 420, Y: 240} coordinates',
    // The Dialog shows within the target element
    target: document.getElementById("container"),
    // Dialog height
    height: '120px',
    // Dialog Width
    width: '360px',
    position: { X: 420, Y: 240 }
});
// Render initialized inner Dialog
secondDialog.appendTo('#secondDialog');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Dialog positioning</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">

```

```

    <meta name="description" content="TypeScript UI Components">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="firstDialog"></div>
        <div id="secondDialog"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Prevent closing of modal dialog in ##Platform_Name## Dialog control

You can prevent closing of modal dialog by setting the [beforeClose](#) event argument cancel value to true.

In the following sample, the dialog is closed when you enter the username value with minimum 4 characters. Otherwise, it will not be closed.

INDEX.JS

```

// To get the all input fields and its container.
var inputElement = document.querySelectorAll('.e-input-group .e-input, .e-
float-input.e-input-group input');
// Add 'e-input-focus' class to the input for achive ripple effect when
focus on the input field.
for (var i = 0; i < inputElement.length; i++) {
    inputElement[i].addEventListener("focus", function () {
        var parentElement = this.parentNode;
        if (parentElement.classList.contains('e-input-in-wrap')) {
            parentElement.parentNode.classList.add('e-input-focus');
        }
        else {

```

```

        this.parentNode.classList.add('e-input-focus');
    }
});
inputElement[i].addEventListener("blur", function () {
    var parentElement = this.parentNode;
    if (parentElement.classList.contains('e-input-in-wrap')) {
        parentElement.parentNode.classList.remove('e-input-focus');
    }
    else {
        this.parentNode.classList.remove('e-input-focus');
    }
});
});
}
// Add 'e-input-btn-ripple' class to the icon element for achive ripple
effect when click on the icon.
var inputIcon = document.querySelectorAll('.e-input-group-icon');
for (var i = 0; i < inputIcon.length; i++) {
    inputIcon[i].addEventListener('mousedown', function () {
        this.classList.add('e-input-btn-ripple');
    });
    inputIcon[i].addEventListener('mouseup', function () {
        var element = this;
        setTimeout(function () {
            element.classList.remove('e-input-btn-ripple');
        }, 500);
    });
}
ej.base.enableRipple(true);
// Initialize Dialog component
var dialog = new ej.popups.Dialog({
    // Enables the header
    header: 'Sign in',
    // Dialog content
    content: document.getElementById("dlgContent"),
    // Enables the footer buttons
    buttons: [{
        // Accessing button component properties by buttonModel property
        buttonModel: {
            //Enables the primary button
            isPrimary: true,
            content: 'Log in',
            cssClass: 'e-primary',
        },
        // Click the footer buttons to hide the Dialog
        click: function () {
            this.hide();
        }
    }],
    // The Dialog shows within the target element
    target: document.getElementById("container"),
    // Dialog width
    width: '300px',
    beforeClose: validation,
    isModal: true,
    beforeOpen: onBeforeopen
});
// Render initialized Dialog

```

```

dialog.appendTo('#dialog');
document.getElementById('targetButton').onclick = function () {
    dialog.show();
    document.getElementById("textvalue").value = "";
    document.getElementById("textvalue2").value = "";
};
function validation(args) {
    let text = document.getElementById('textvalue');
    let text1 = document.getElementById('textvalue2');
    if (text.value === "" && text1.value === "") {
        args.cancel= true;
        alert("Enter the username and password")
    } else if (text.value === "") {
        args.cancel= true;
        alert("Enter the username")
    } else if (text1.value === "") {
        args.cancel= true;
        alert("Enter the password")
    } else if (text.value.length < 4) {
        args.cancel= true;
        alert("Username must be minimum 4 characters")
    } else {
        args.cancel= false;
        document.getElementById("textvalue").value = "";
        document.getElementById("textvalue2").value = "";
    }
}
function onBeforeopen() {
    document.getElementById('dlgContent').style.visibility = 'visible';
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Dialog</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="TypeScript UI Components">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>

```



```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button class="e-control e-btn" id="targetButton" role="button" e-
ripple="true" style="position: absolute;">Open Dialog</button>
        <div id="dialog"></div>
        <div id="dlgContent" style="visibility: hidden" class="wrap">
            <div id="input-container">
                <div class="e-float-input">
                    <input id="textvalue" type="text" required="">
                    <span class="e-float-line"></span>
                    <label class="e-float-text">Username</label>
                </div>
            </div>
            <div class="form-group">
                <div class="e-float-input">
                    <input id="textvalue2" type="Password" required="">
                    <span class="e-float-line"></span>
                    <label class="e-float-text">Password</label>
                </div>
            </div>
        </div>
    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Prevent the focus on the first element in ##Platform_Name## Dialog control

By default, the dialog focuses on the first elements of the content area which can be active and focusable. You can prevent this default focusing behavior using the [open](#) event and by enabling the `preventFocus` argument.

Bind the open event and enable the `preventFocus` argument within an event like the below sample.

INDEX.JS

```

ej.base.enableRipple(true);
// Initialization of Dialog component
var dialog = new ej.popups.Dialog({
    header: "Sign In",
    buttons: [{ buttonModel: { isPrimary: true, content: 'Yes' }, click:
btnClick }, { buttonModel: { content: 'No' }, click: btnClick }],
    target: document.getElementById("container"),
    height: 'auto',
    width: '300px',
    open: onOpen
});
// Render initialized Dialog

```

```

dialog.appendTo('#dialog');
// Sample level code to handle the button click action
document.getElementById('targetButton').onclick = function () {
    // Call the show method to open the Dialog
    dialog.show();
};
function btnClick() {
    dialog.hide();
}
function onOpen(args: OpenEventArgs) {
    //Preventing the default dialog focus
    args.preventDefault = true;
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Dialog with header</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="TypeScript UI Components">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button class="e-control e-btn" id="targetButton" role="button" e-
ripple="true" style="position: absolute;">Open Dialog</button>
        <div id="dialog">
            <div class="form-group"><label for="email">Email:</label>
                <input type="email" class="form-control" id="email">
            </div>
            <div class="form-group">
                <label for="comment">Password:</label>
                <input type="password" class="form-control" id="password">
            </div>
        </div>
    </div>

```

```
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Prevent opening of the dialog in ##Platform_Name## Dialog control

You can prevent opening of the dialog by setting the [beforeOpen](#) event argument cancel value to true.

In the following sample, the success dialog is opened when you enter the username value with minimum 4 characters. Otherwise, it will not be opened.

INDEX.JS

```
// To get the all input fields and its container.
var inputElement = document.querySelectorAll('.e-input-group .e-input, .e-
float-input.e-input-group input');
// Add 'e-input-focus' class to the input for achive ripple effect when
focus on the input field.
for (var i = 0; i < inputElement.length; i++) {
  inputElement[i].addEventListener("focus", function () {
    var parentElement = this.parentNode;
    if (parentElement.classList.contains('e-input-in-wrap')) {
      parentElement.parentNode.classList.add('e-input-focus');
    }
    else {
      this.parentNode.classList.add('e-input-focus');
    }
  });
  inputElement[i].addEventListener("blur", function () {
    var parentElement = this.parentNode;
    if (parentElement.classList.contains('e-input-in-wrap')) {
      parentElement.parentNode.classList.remove('e-input-focus');
    }
    else {
      this.parentNode.classList.remove('e-input-focus');
    }
  });
}
// Add 'e-input-btn-ripple' class to the icon element for achive ripple
effect when click on the icon.
var inputIcon = document.querySelectorAll('.e-input-group-icon');
for (var i = 0; i < inputIcon.length; i++) {
  inputIcon[i].addEventListener('mousedown', function () {
    this.classList.add('e-input-btn-ripple');
  });
  inputIcon[i].addEventListener('mouseup', function () {
    var element = this;
    setTimeout(function () {
      element.classList.remove('e-input-btn-ripple');
    }, 500);
  });
}
```

```

ej.base.enableRipple(true);
// Initialize Dialog component
var dialog = new ej.popups.Dialog({
  header: 'Success',
  buttons: [
    {
      // Click the footer buttons to hide the Dialog
      'click': function () {
        dialog.hide();
      },
      // Accessing button component properties by buttonModel property
      buttonModel: {
        //Enables the primary button
        isPrimary: true,
        cssClass: 'e-flat',
        content: 'Dismiss'
      }
    }
  ],
  // Dialog content
  content: 'Congratulations! Login Success',
  // The Dialog shows within the target element
  target: document.getElementById("container"),
  // Dialog width
  width: '280px',
  isModal: true,
  visible: false,
  beforeOpen: validation
});
// Render initialized Dialog
dialog.appendTo('#dialog');
document.getElementById('targetButton').onclick = function () {
  dialog.show();
};
function validation(args) {
  let text = document.getElementById('textvalue');
  let text1 = document.getElementById('textvalue2');
  if (text.value === "" && text1.value === "") {
    args.cancel= true;
    alert("Enter the username and password")
  } else if (text.value === "") {
    args.cancel= true;
    alert("Enter the username")
  } else if (text1.value === "") {
    args.cancel= true;
    alert("Enter the password")
  } else if (text.value.length < 4) {
    args.cancel= true;
    alert("Username must be minimum 4 characters")
  } else {
    args.cancel= false;
    document.getElementById("textvalue").value = "";
    document.getElementById("textvalue2").value = "";
  }
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Dialog</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript UI Components">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="login-form">
      <div class="wrap">
        <div id="heading">Sign in</div>
        <div id="input-container">
          <div class="e-float-input e-input-group">
            <input id="textvalue" type="text" required="">
            <span class="e-float-line"></span>
            <label class="e-float-text">Username</label>
          </div>
          <div class="e-float-input e-input-group">
            <input id="textvalue2" type="password" required="">
            <span class="e-float-line"></span>
            <label class="e-float-text">Password</label>
          </div>
        </div>
        <div class="button-contain">
          <button class="e-control e-btn e-info" id="targetButton"
role="button" e-ripple="true">Log in</button>
        </div>
      </div>
    </div>
    <div id="dialog"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";

```

```

}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Read all the values from dialog on button click in `##Platform_Name##` Dialog control

You can read the dialog element values by binding the action handler to the footer buttons. The buttons property provides the options to bind events to the action buttons. For detailed information about buttons, refer to the [footer](#) section. In the below sample, value of input elements within the dialog has been checked in the footer button click event and send the values as the content of confirmation dialog.

INDEX.JS

```

ej.base.enableRipple(true);
var dlgObj = new ej.popups.Dialog({
  header: 'User details',
  content: document.getElementById("dlgContent"),
  showCloseIcon: true,
  visible: false,
  buttons: [{
    buttonModel: { isPrimary: true, content: 'Submit' }, click:
function() {
  createModalDialog();
},
}],
target: document.querySelector('body'),
width: '400px',
animationSettings: { effect: 'Zoom' },
beforeOpen: onBeforeopen
});
dlgObj.appendTo('#dialog');
var button = new ej.buttons.Button({});
button.appendTo('#dialogButton');
document.getElementById('openBtn').onclick = function () {
  dlgObj.show();
};
var modalObj;
function createModalDialog() {
  dlgObj.hide()
  if (!document.getElementById('modalDialog').classList.contains('e-dialog')) {
    modalObj = new ej.popups.Dialog({
      header: 'User details',
      content: getDynamicContent(),
      showCloseIcon: true,
      isModal: true,
      visible: true,
      width: '600px',
      buttons: [{buttonModel: {isPrimary: true, content: 'Yes'}, click:
function() {
        this.hide();
      }}, {buttonModel: {isPrimary: false, content: 'No'}, click:
function() {
        this.hide();
        dlgObj.show();
      }
}],
}
  }
}

```

```

        target: document.querySelector('body'),
        animationSettings: { effect: 'Zoom' }
    });
    modalObj.appendTo('#modalDialog');

    } else {
        modalObj.content = getDynamicContent();
        modalObj.show()
    }
}
function getDynamicContent() {
    let input = document.getElementById('dialog').querySelector('#name');
    let email = document.getElementById('dialog').querySelector('#email');
    let contact =
document.getElementById('dialog').querySelector('#contact');
    let address =
document.getElementById('dialog').querySelector('#address');
    let template = "<div class='row'><div class='col-xs-6 col-sm-6 col-lg-6 col-md-6'><b>Confirm your details</b></div>" +
        "</div><div class='row'><div class='col-xs-6 col-sm-6 col-lg-6 col-md-6'><span id='name'> Name: </span>" +
        "</div><div class='col-xs-6 col-sm-6 col-lg-6 col-md-6'><span id='nameValue'>"+ input.value + "</span> </div></div>" +
        "<div class='row'><div class='col-xs-6 col-sm-6 col-lg-6 col-md-6'><span id='email'> Email: </span>" +
        "</div><div class='col-xs-6 col-sm-6 col-lg-6 col-md-6'><span id='emailValue'>"+ email.value +
        "</span></div></div><div class='row'><div class='col-xs-6 col-sm-6 col-lg-6 col-md-6'>"+
        "<span id='Contact'> Contact number: </span></div><div class='col-xs-6 col-sm-6 col-lg-6 col-md-6'>"+
        "<span id='contactValue'>"+ contact.value + "</span></div></div><div class='row'><div class='col-xs-6 col-sm-6 col-lg-6 col-md-6'>"+
        "<span id='Address'> Address: </span> </div><div class='col-xs-6 col-sm-6 col-lg-6 col-md-6'><span id='AddressValue'>"+ address.value
        + "</span></div></div>"
    return template;
}
function onBeforeopen() {
    document.getElementById('dlgContent').style.visibility = 'visible';
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Dialog</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="TypeScript UI Components">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="target">
            <center><button id="openBtn" class="e-control e-
btn">Open</button></center>
            <div id="dialog"></div>
            <div id="modalDialog"></div>
            <form id="dlgContent" style="visibility: hidden">
                <div class="form-group">
                    <label for="name">Name:</label>
                    <input type="name" value="" class="form-control"
id="name">
                </div>
                <div class="form-group">
                    <label for="email">Email Id:</label>
                    <input type="email" value="user@syncfusion.com"
class="form-control" id="email">
                </div>
                <div class="form-group">
                    <label for="contact">Contact Number:</label>
                    <input type="contact" class="form-control" id="contact">
                </div>
                <div class="form-group">
                    <label for="address">Address:</label>
                    <textarea class="form-control" rows="5"
id="address"></textarea>
                </div>
            </form>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```


Customize the dialog appearance in ##Platform_Name## Dialog control

You can customize the dialog appearance by providing dialog template as string or HTML element to the [content](#) property. In the following sample, dialog is customized as error window appearance.

INDEX.JS

```
ej.base.enableRipple(true);
var dlgObj = new ej.popups.Dialog({
  header: 'File and Folder Rename',
  content: document.getElementById("dlgContent"),
  showCloseIcon: true,
  visible: false,
  buttons: [{
    buttonModel: { isPrimary: true, content: 'close' }, click:
function() { this.hide() }
  }],
  target: document.querySelector('body'),
  width: '400px',
  animationSettings: { effect: 'Zoom' },
  beforeOpen: onBeforeopen
});
dlgObj.appendTo('#dialog');
var button = new ej.buttons.Button({});
button.appendTo('#dialogButton');
document.getElementById('openBtn').onclick = function () {
  dlgObj.show();
};
function onBeforeopen() {
  document.getElementById('dlgContent').style.visibility = 'visible';
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Dialog customization</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript UI Components">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/bootstrap.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/bootstrap.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/bootstrap.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="target">
            <center><button id="openBtn" class="e-control e-
btn">Open</button></center>
            <div id="dialog"></div>
            <div id="dlgContent" style="visibility: hidden" class="dialog-
content">

                <div class="msg-wrapper col-lg-12">
                    <span class="e-icons close-icon col-lg-2"></span>
                    <span class="error-msg col-lg-10">
                        Can not rename 'pictures' because a file or folder
with that name already exists
                    </span>
                </div>
                <div class="error-detail col-lg-8">
                    <span>Specify a different name</span>
                </div>
            </div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Close dialog while click on outside of dialog in ##Platform_Name## Dialog control

By default, dialog can be closed by pressing Esc key and clicking the close icon on the right of dialog header. It can also be closed by clicking outside of the dialog using `hide` method. Set the [CloseOnEscape](#) property value to false to prevent closing of the dialog when pressing Esc key.

In the following sample, dialog is closed when clicking outside the dialog area using [hide](#) method.

INDEX.TS

```

import { enableRipple } from '@syncfusion/ej2-base';
import { Button } from '@syncfusion/ej2-buttons';
import { Dialog } from '@syncfusion/ej2-popups';
let dialogObj: Dialog = new Dialog({
    header: 'Delete Multiple Items',
    content: "Are you sure you want to permanently delete all of these
items?",
    showCloseIcon: true,
    buttons: [{ buttonModel: { isPrimary: true, content: 'Yes' }, click:
btnClick }, { buttonModel: { content: 'No' }, click: btnClick }],
    target: document.body,
    height: 'auto',
    width: '300px',

```

```

        animationSettings: { effect: 'Zoom' },
        closeOnEscape: true
    });
    dialogObj.appendTo('#dialog');
    document.getElementById('openBtn').onclick = (): void => {
        dialogObj.show();
    };
    function btnClick() {
        dialogObj.hide();
    }
    document.onclick = (args: any) : void => {
        if(args.target.id === 'target') {
            dialogObj.hide();
        }
    }
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Dialog</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="TypeScript UI Components">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body class="close-dialog">

    <div id="container">
        <div id="target" class="close-dialog">
            <center><button id="openBtn" class="e-control e-
btn">Open</button></center>
            <div id="dialog"> </div>
        </div>
    </div>
</body>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}

```

```

}
    </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Add an icons to dialog buttons in `##Platform_Name##` Dialog control

You can add icons to the dialog buttons using the [buttons](#) property or [footerTemplate](#) property . For detailed information about dialog buttons, refer to the [documentation](#) section.

In the following sample, dialog footer buttons are customized with icons using `buttons` property.

INDEX.JS

```

ej.base.enableRipple(true);
var dialogObj = new ej.popups.Dialog({
  header: 'Delete Multiple Items',
  content: "Are you sure you want to permanently delete all of these items?",
  showCloseIcon: true,
  buttons: [{ buttonModel: { isPrimary: true, content: 'Yes', iconCss: 'e-icons e-ok-icon' }, click: btnClick }, { buttonModel: { content: 'No', iconCss: 'e-icons e-close-icon' }, click: btnClick }],
  target: document.body,
  height: 'auto',
  width: '300px',
  animationSettings: { effect: 'Zoom' },
  closeOnEscape: true
});
dialogObj.appendTo('#dialog');
document.getElementById('openBtn').onclick = function () {
  dialogObj.show();
};
function btnClick() {
  dialogObj.hide();
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Dialog button with icons</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript UI Components">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body class="close-dialog">

    <div id="container">
        <div id="target" class="close-dialog">
            <center><button id="openBtn" class="e-control e-
btn">Open</button></center>
            <div id="dialog"> </div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

In the following sample, dialog footer buttons are customized with icons using `footerTemplate` property.

INDEX.JS

```

ej.base.enableRipple(true);
var dialogObj = new ej.popups.Dialog({
    header: 'Delete Multiple Items',
    content: "Are you sure you want to permanently delete all of these
items?",
    showCloseIcon: true,
    footerTemplate: '<button id="Button1" class="e-control e-btn e-primary
e-flat" data-ripple="true"><span class="e-btn-icon e-icons e-ok-icon e-icon-
left"></span>Yes</button><button id="Button2" class="e-control e-btn e-flat"
data-ripple="true"><span class="e-btn-icon e-icons e-close-icon e-icon-
left"></span>No</button>',
    target: document.body,
    height: 'auto',
    width: '300px',
    animationSettings: { effect: 'Zoom' },
    closeOnEscape: true
});
dialogObj.appendTo('#dialog');
document.getElementById('openBtn').onclick = function () {
    dialogObj.show();
};
document.getElementById('Button1').onclick = function () {
    dialogObj.hide();
};
document.getElementById('Button2').onclick = function () {
    dialogObj.hide();
};

```

```
};
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Dialog button with icons</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript UI Components">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body class="close-dialog">

  <div id="container">
    <div id="target" class="close-dialog">
      <center><button id="openBtn" class="e-control e-
btn">Open</button></center>
      <div id="dialog"> </div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Add a minimize maximize buttons in ##Platform_Name## Dialog control

Dialog allows end users to either minimize or maximize the Dialog component. You can add minimize and maximize custom buttons near the close icon in the Dialog header using the [headerTemplate](#) property and handle the actions in the button click events, as shown in the following sample.

INDEX.JS

```
ej.base.enableRipple(true);
```

```

var dialogObj = new ej.popups.Dialog({
  header: `<span class='title'>Dialog</span>
  <span class='e-icons sf-icon-Maximize' id='max-btn'
title='Maximize'></span>
  <span class='e-icons sf-icon-Minimize' id='min-btn'
title='Minimize'></span>`,
  content: "This is a dialog with minimize and maximize buttons",
  showCloseIcon: true,
  buttons: [{ buttonModel: { isPrimary: true, content: 'Yes', iconCss: 'e-
icons e-ok-icon' }, click: btnClick }, { buttonModel: { content: 'No',
iconCss: 'e-icons e-close-icon' }, click: btnClick }],
  target: document.body,
  height: 'auto',
  width: '300px',
  animationSettings: { effect: 'Zoom' },
  closeOnEscape: true
});
dialogObj.appendTo('#dialog');
document.getElementById('openBtn').onclick = function () {
  dialogObj.show();
};
function btnClick() {
  dialogObj.hide();
}
let hide;
let isFullScreen;
let dialogOldPositions;
document.getElementById("max-btn").addEventListener("click", function() {
  let maximizeIcon;
  if (dialogObj.element.classList.contains('dialog-minimized')) {
    dialogObj.element.querySelector('#min-btn').classList.add('sf-icon-
Minimize');
    dialogObj.element.querySelector('#min-btn').classList.remove('sf-
icon-Restore');
    dialogObj.element.querySelector('#min-btn').setAttribute('title',
'Minimize');
  }
  if (!dialogObj.element.classList.contains('dialog-maximized') &&
!isFullScreen) {
    maximizeIcon = dialogObj.element.querySelector(".e-dlg-header-
content .sf-icon-Maximize");
  } else {
    maximizeIcon = dialogObj.element.querySelector(".e-dlg-header-
content .sf-icon-Restore");
  }
  if (!dialogObj.element.classList.contains('dialog-maximized')) {
    dialogObj.element.classList.add('dialog-maximized');
    dialogObj.show(true);
    maximizeIcon.classList.add('sf-icon-Restore');
    maximizeIcon.setAttribute('title', 'Restore');
    maximizeIcon.classList.remove('sf-icon-Maximize');
    dialogObj.element.querySelector('.e-dlg-
content').classList.remove('hide-content');
    isFullScreen = true;
  } else {
    dialogObj.element.classList.remove('dialog-maximized');
    dialogObj.show(false);
  }
});

```

```

        maximizeIcon.classList.remove('sf-icon-Restore');
        maximizeIcon.classList.add('sf-icon-Maximize');
        maximizeIcon.setAttribute('title', 'Maximize');
        dialogObj.element.querySelector('.e-dlg-
content').classList.remove('hide-content');
        dialogObj.position = dialogOldPositions;
        dialogObj.dataBind();
        isFullScreen = false;
    }
});

document.getElementById("min-btn").addEventListener("click", function()
{
    let minimizeIcon = dialogObj.element.querySelector(".e-dlg-header-
content .sf-icon-Minimize");
    if (!dialogObj.element.classList.contains('e-dlg-fullscreen')) {
        if (!dialogObj.element.classList.contains('dialog-minimized')) {
            dialogOldPositions = { X: dialogObj.position.X, Y:
dialogObj.position.Y }
            dialogObj.element.classList.add('dialog-minimized');
            dialogObj.element.classList.remove('dialog-maximized');
            dialogObj.element.querySelector('.e-dlg-
content').classList.add('hide-content');
            dialogObj.position = { X: 'center', Y: 'bottom' };
            dialogObj.dataBind();
            minimizeIcon.classList.add('sf-icon-Restore');
            minimizeIcon.setAttribute('title', 'Restore');
        } else {
            dialogObj.element.classList.remove('dialog-minimized');
            dialogObj.element.querySelector('.e-dlg-
content').classList.remove('hide-content');
            minimizeIcon.classList.add('sf-icon-Minimize');
            minimizeIcon.setAttribute('title', 'Minimize');
            minimizeIcon.classList.remove('sf-icon-Restore');
            dialogObj.position = dialogOldPositions;
            dialogObj.dataBind();
        }
    } else {
        dialogObj.show(false);
        dialogObj.element.classList.remove('dialog-maximized');
        dialogObj.element.classList.add('dialog-minimized');
        dialogObj.element.querySelector('.e-dlg-
content').classList.add('hide-content');
        minimizeIcon.classList.remove('sf-icon-Minimize');
        minimizeIcon.removeAttribute('title');
        dialogObj.position = { X: 'center', Y: 'bottom' };
        dialogObj.dataBind();
        isFullScreen = true;
    }
});

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
<title>Essential JS 2 Dialog button with icons</title>

```



```

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="TypeScript UI Components">
<meta name="author" content="Syncfusion">
<link href="styles.css" rel="stylesheet">
<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body class="close-dialog">

    <div id="container">
        <div id="target" class="close-dialog">
            <center><button id="openBtn" class="e-control e-
btn">Open</button></center>
            <div id="dialog"> </div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Setting max height to the dialog in ##Platform_Name## Dialog control

By default, the maxHeight for the Dialog is calculated based on the target. If the target is not specified externally, the Dialog consider the body as target and will calculate the maxHeight based on it. We have an option to set the maxHeight of the Dialog in the [beforeOpen](#) event.

INDEX.JS

```

ej.base.enableRipple(true);
// Initialize Dialog component.
var dialog = new ej.popups.Dialog({
    width: '800px',
    showCloseIcon: true,
    position: { X: 'center', Y: 'center' },
    header: 'Dialog',
    created: onCreate,

```

```

    beforeOpen: onOpen,
    // The Dialog shows within the target element
    target: document.getElementById("container"),
    visible: false,
  });
  // Render initialized Dialog.
  dialog.appendTo('#dialog');
  // Sample level code to handle the button click action.
  document.getElementById('targetButton').onclick = function () {
    // Call the show method to open the Dialog.
    dialog.show();
  }
  function onCreate() {
    document.getElementById('dlgContent').style.display = 'block';
    dialog.refreshPosition();
  }
  function onOpen(args) {
    // setting maxHeight to the Dialog.
    args.maxHeight = '300px';
  }
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Dialog with scrollable content</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript UI Components">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <button class="e-control e-btn" id="targetButton" role="button" e-
ripple="true" style="position: absolute;">Open Dialog</button>
    <div id="dialog">
      <div id="dlgContent" style="display: none">
        <div>
          <b>JavaScript:</b><br>
          JavaScript is a high-level, dynamic, untyped, and
          interpreted programming language. It has been standardized in the ECMAScript
          language specification. Alongside HTML and CSS, it is
          one of the three essential technologies of World Wide Web

```

content production; the majority of websites employ it and it **is** supported by all modern Web browsers without plug-ins. JavaScript **is** a prototype-based programming language with first-**class functions**, making it a multi-paradigm language, supporting **object-oriented** , imperative, and functional programming styles.

MVC:

Model-view-controller (MVC) **is** a software architecture pattern which separates the representation of information **from** the user's interaction with it. The model consists of application data, business rules, logic, and functions. A view can be any output representation of data, such **as** a chart or a diagram. Multiple views of the same data are possible, such **as** a bar chart **for** management and a tabular view **for** accountants. The controller mediates input, converting it to commands **for** the model or view. The central ideas behind MVC are code reusability and **in** addition to dividing the application **into** three kinds of components, the MVC design defines the interactions between them.

A controller can send commands to its associated view to change the view's presentation of the model (e.g., by scrolling through a document). It can also send commands to the model to update the model's state (e.g., editing a document).

A model notifies its associated views and controllers when there **is** a change **in** its state. This notification allows the views to produce updated output, and the controllers to change the available **set** of commands. A passive implementation of MVC omits these notifications because the application does not require them or the software platform does not support them.

A view requests **from** the model the information that it needs to generate an output representation to the user.

</div>

</div>

</div>

</div>

<script>

var ele = document.getElementById('container');

if(ele) {

 ele.style.visibility = "visible";

}

</script>

<script src="index.js" type="text/javascript"></script>

</body></html>

DocumentEditor

Overview

The Document Editor component is used to create, edit, view, and print Word documents in web applications. All the user interactions and editing operations that run purely in the client-side provides much faster editing experience to the users.

Key Features

- [Opens](#) the native Syncfusion Document Text (*.sfdt) format documents in the client-side.

- [Saves the documents](#) in the client-side as **Syncfusion Document Text (.sfdt) and Word document (.docx)**.
- Supports document elements like text, [image](#), [table](#), fields, [bookmark,shapes](#), [section](#), [header and footer](#).
- Supports the commonly used fields like [hyperlink](#), page number, page count, and table of contents.
- Supports formats like [text](#), [paragraph](#), [bullets and numbering](#), [table](#), [page settings](#), etc.
- Provides support to create, edit, and apply [paragraph and character styles](#).
- Provides support to [find and replace](#) text within the document.
- Supports all the common editing and formatting operations along with [undo and redo](#).
- Provides support to [cut](#), [copy](#), and [paste](#) rich text contents within the component. Also allows pasting simple text to and from other applications.
- Provides support to insert, and edit [form fields](#).
- Provides support to insert, and edit [comments](#).
- Provides support to track the [inserted and deleted content](#).
- Provides support to perform [spell checking](#) for any input text
- Allows user interactions like [zoom](#), [scroll](#), select contents through touch, mouse, and keyboard.
- Provides intuitive UI options like context menu, [dialogs](#), and [navigation pane](#).
- [Localizes](#) all the static text to any desired language.
- Allows to create a lightweight Word viewer using module injection to view and [prints](#) Word documents.
- Provides a [server-side helper library](#) to open the Word documents like DOCX, DOC, WordML, RTF, and Text, by converting it to SFDT file format.

Supported Web platforms

Other platforms

- [Javascript](#)
- [Angular](#)
- [React](#)
- [Vue](#)
- [ASP.NET Core](#)
- [ASP.NET MVC](#)
- [Blazor](#)

Supported platforms for server-side dependencies

You can deploy web APIs for server-side dependencies of Document Editor component in the following platforms.

- [ASP.NET Core](#)
- [ASP.NET MVC](#)
- [Java](#)

To know more about server-side dependencies, refer this [page](#).

Which operations require server-side interaction

- Open file formats other than SFDT

- Paste with formatting
- Restrict editing
- Spellcheck
- Save as file formats other than SFDT and DOCX

Note: If you don't require the above functionalities then you can deploy as pure client-side component without any server-side interactions.

Feature module in ##Platform_Name## Document editor control

Document Editor features are segregated into individual feature-wise modules to enable selective referencing. By default, the Document Editor displays the document in read-only mode. The required modules should be injected to extend its functionality. The following are the selective modules of Document Editor that can be included as required:

- **Print** - Prints the document.
- **SfdtExport** - Exports the document as Syncfusion Document Text (.SFDT) file.
- **Selection** - Selects a portion of the document and copy it to the clipboard.
- **Search** - Searches specific text and navigate between the results.
- **WordExport** - Exports the document as Word Document (.DOCX) file.
- **TextExport** - Exports the document as Text Document (.TXT) file.
- **Editor** - Performs all kind of editing operations.
- **EditorHistory** - Maintains the history of editing operations so that you can perform undo and redo at any time.
- User interface options such as context menu, options pane, image resizer, and dialog are available as individual modules.

In addition to injecting the required modules in your application, enable corresponding properties to extend the functionality for a Document Editor instance.

Refer to the following table.

Module	Dependent modules to be injected for extending the functionality of Document Editor in your application	Property to enable the functionality for a Document Editor instance
--------	---	---

---	---	---
-----	-----	-----

Print	DocumentEditor.Inject(Print)	let documenteditor: DocumentEditor = new DocumentEditor({ enablePrint: true });
-------	------------------------------	---

SfdtExport	DocumentEditor.Inject(SfdtExport)	let documenteditor: DocumentEditor = new DocumentEditor({ enableSfdtExport: true });
------------	-----------------------------------	--

Selection	DocumentEditor.Inject(Selection)	let documenteditor: DocumentEditor = new DocumentEditor({ enableSelection: true });
-----------	----------------------------------	---

Search	DocumentEditor.Inject(Selection, Search)	let documenteditor: DocumentEditor = new DocumentEditor({ enableSearch: true });
--------	--	--

WordExport	DocumentEditor.Inject(SfdtExport, WordExport)	let documenteditor: DocumentEditor = new DocumentEditor({ enableWordExport: true });
------------	---	--

```
|TextExport| DocumentEditor.Inject(SfdtExport, TextExport)|let documenteditor:
DocumentEditor = new DocumentEditor({ enableTextExport: true });|

|Editor| DocumentEditor.Inject(Selection, Editor)|let documenteditor: DocumentEditor = new
DocumentEditor({ isReadOnly: false, enableEditor: true });|

|EditorHistory| DocumentEditor.Inject(Selection, Editor, EditorHistory)|let documenteditor:
DocumentEditor = new DocumentEditor({ isReadOnly: false, enableEditor: true,
enableEditorHistory: true });|

|OptionsPane(Find)| DocumentEditor.Inject(Selection, Search, OptionsPane)|let documenteditor:
DocumentEditor = new DocumentEditor({ enableSearch: true, enableOptionsPane: true });|

|OptionsPane(Find and Replace)| DocumentEditor.Inject(Selection, Search, Editor,
OptionsPane)|let documenteditor: DocumentEditor = new DocumentEditor({ isReadOnly: false,
enableEditor: true, enableSearch: true, enableOptionsPane: true });|

|ContextMenu| DocumentEditor.Inject(Selection, ContextMenu)|let documenteditor:
DocumentEditor = new DocumentEditor({ enableSelection: true, enableContextMenu: true });|

|ImageResizer| DocumentEditor.Inject(Selection, Editor, ImageResizer)|let documenteditor:
DocumentEditor = new DocumentEditor({ isReadOnly: false, enableEditor: true,
enableImageResizer: true });|

|HyperlinkDialog| DocumentEditor.Inject(Selection, Editor, HyperlinkDialog)|let documenteditor:
DocumentEditor = new DocumentEditor({ isReadOnly: false, enableEditor: true,
enableHyperlinkDialog: true });|

|TableDialog| DocumentEditor.Inject(Selection, Editor, TableDialog)|let documenteditor:
DocumentEditor = new DocumentEditor({ isReadOnly: false, enableEditor: true,
enableTableDialog: true });|

|FontDialog| DocumentEditor.Inject(Selection, Editor, FontDialog)|let documenteditor:
DocumentEditor = new DocumentEditor({ isReadOnly: false, enableEditor: true,
enableFontDialog: true });|

|ParagraphDialog| DocumentEditor.Inject(Selection, Editor, ParagraphDialog)|let
documenteditor: DocumentEditor = new DocumentEditor({ isReadOnly: false, enableEditor:
true, enableParagraphDialog: true });|

|BookmarkDialog| DocumentEditor.Inject(Selection, Editor, BookmarkDialog)|let
documenteditor: DocumentEditor = new DocumentEditor({ isReadOnly: false, enableEditor:
true, enableBookmarkDialog: true });|

|PageSetupDialog| DocumentEditor.Inject(Selection, Editor, PageSetupDialog)|let
documenteditor: DocumentEditor = new DocumentEditor({ isReadOnly: false, enableEditor:
true, enablePageSetupDialog: true });|

|TableOfContentsDialog| DocumentEditor.Inject(Selection, Editor, TableOfContentsDialog)|let
documenteditor: DocumentEditor = new DocumentEditor({ isReadOnly: false, enableEditor:
true, enableTableOfContentsDialog: true });|
```

```
|ListDialog|DocumentEditor.Inject(Selection, Editor, ListDialog)|let documenteditor:
DocumentEditor = new DocumentEditor({ isReadOnly: false, enableEditor: true,
enableListDialog: true });|
```

```
|TablePropertiesDialog|DocumentEditor.Inject(Selection, Editor, TablePropertiesDialog)|let
documenteditor: DocumentEditor = new DocumentEditor({ isReadOnly: false, enableEditor:
true, enableTablePropertiesDialog: true });|
```

```
|CellOptionsDialog|DocumentEditor.Inject(Selection, Editor, CellOptionsDialog)|let
documenteditor: DocumentEditor = new DocumentEditor({ isReadOnly: false, enableEditor:
true, enableTablePropertiesDialog: true });|
```

```
|BordersAndShadingDialog|DocumentEditor.Inject(Selection, Editor,
BordersAndShadingDialog)|let documenteditor: DocumentEditor = new DocumentEditor({
isReadOnly: false, enableEditor: true, enableBordersAndShadingDialog: true });|
```

```
|TableOptionsDialog|DocumentEditor.Inject(Selection, Editor, TableOptionsDialog)|let
documenteditor: DocumentEditor = new DocumentEditor({ isReadOnly: false, enableEditor:
true, enableTableOptionsDialog: true });|
```

```
|StylesDialog|DocumentEditor.Inject(Selection, Editor, StylesDialog,StyleDialog)|let
documenteditor: DocumentEditor = new DocumentEditor({ isReadOnly: false, enableEditor:
true, enableStyleDialog: true ,enableStylesDialog: true });|
```

```
|StyleDialog|DocumentEditor.Inject(Selection, Editor, StyleDialog)|let documenteditor:
DocumentEditor = new DocumentEditor({ isReadOnly: false, enableEditor: true,
enableStyleDialog: true });|
```

```
|BulletsAndNumberingDialog|DocumentEditor.Inject(Selection, Editor,
BulletsAndNumberingDialog)|let documenteditor: DocumentEditor = new DocumentEditor({
isReadOnly: false, enableEditor: true, enableStyleDialog: true });|
```

[Import in ##Platform_Name## Document editor control](#)

In Document Editor, the documents are stored in its own format called **Syncfusion Document Text (SFDT)**.

The following example shows how to open SFDT data in Document Editor.

INDEX.JS

```
var documenteditor = new ej.documenteditor.DocumentEditor({ });
documenteditor.appendTo('#DocumentEditor');
var sfdt = {
  "sections": [
    {
      "blocks": [
        {
          "inlines": [
            {
              "characterFormat": {
                "bold": true,
                "italic": true
              },
              "text": "Hello World"
            }
          ]
        }
      ]
    }
  ]
}
```

```

    }
  ]
},
"headersFooters": {
}
}
];
};
documenteditor.open(JSON.stringify(sfdt));

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
documenteditor/styles/fabric.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div style="width:100%;height: 100%">
      <!--Element which will render as DocumentEditor -->
      <div id="DocumentEditor"></div>
    </div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Import document from local machine

The following example shows how to import document from local machine.

INDEX.JS

```

var documenteditor = new ej.documenteditor.DocumentEditor({ });
documenteditor.appendTo('#DocumentEditor');
document.getElementById('file_upload').setAttribute('accept', '.sfdt');
document.getElementById("import").addEventListener("click", function () {
  document.getElementById('file_upload').click();
});

```



```

});
document.getElementById('file_upload').addEventListener("change", function
(e) {
    if (e.target.files[0]) {
        var file = e.target.files[0];
        if (file.name.substr(file.name.lastIndexOf('.')) === '.sfdt') {
            var fileReader = new FileReader();
            fileReader.onload = function (e) {
                var contents = e.target.result;
                documenteditor.open(contents);
            };
            fileReader.readAsText(file);
            documenteditor.documentName = file.name.substr(0,
file.name.lastIndexOf('.'));
        }
    }
});

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
documenteditor/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <input type="file" id="file_upload" style="position:fixed; left:-100em">
    <div id="container">
        <div>
            <button id="import">Import</button>
        </div>
        <!--Element which will render as DocumentEditor -->
        <div id="DocumentEditor"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Convert word documents into SFDT

You can convert word documents into SFDT format using the .NET Standard library [Syncfusion.EJ2.WordEditor.AspNet.Core](#) by the web API service implementation. This library helps you to convert word documents (.dotx,.docx,.docm,.dot,.doc), rich text format documents (.rtf), and text documents (.txt) into SFDT format.

Note: The Syncfusion Document Editor component's document pagination (page-by-page display) can't be guaranteed for all the Word documents to match the pagination of Microsoft Word application. For more information about [why the document pagination \(page-by-page display\) differs from Microsoft Word](#)

Please refer the following example for converting word documents into SFDT.

```
`ts
import { DocumentEditor } from '@syncfusion/ej2-documenteditor';

// Initialize the Document Editor component.
let documenteditor: DocumentEditor = new DocumentEditor();
documenteditor.appendTo('#DocumentEditor');

document.getElementById('file_upload').setAttribute('accept',
'.dotx,.docx,.docm,.dot,.doc,.rtf,.txt,.xml,.sfdt');

//Open file picker.
document.getElementById("import").addEventListener("click", (): void => {
document.getElementById('file_upload').click();
});

document.getElementById('file_upload').addEventListener("change", (e: any): void => {
if (e.target.files[0]) {
//Get the selected file.
let file = e.target.files[0];
if (file.name.substr(file.name.lastIndexOf('.') != '.sfdt') {
loadFile(file);
}
}
});

function loadFile(file: File): void {
let ajax: XMLHttpRequest = new XMLHttpRequest();
ajax.open('POST', 'https://localhost:4000/api/documenteditor/Import', true);
ajax.onreadystatechange = () => {
if (ajax.readyState === 4) {
if (ajax.status === 200 || ajax.status === 304) {
```

```
//Open SFDT text in Document Editor
documenteditor.open(ajax.responseText);
}
}
}

let formData: FormData = new FormData();
formData.append('files', file);
//Send the selected file to web api for converting it into sfdt.
ajax.send(formData);
}
,
```

Here's how to handle the server-side action for converting word document in to SFDT.

```
`c#
[AcceptVerbs("Post")]
public string Import(Microsoft.AspNetCore.Http.IFormCollection data)
{
    if (data.Files.Count == 0)
        return null;

    System.IO.Stream stream = new System.IO.MemoryStream();
    Microsoft.AspNetCore.Http.IFormFile file = data.Files[0];
    int index = file.FileName.LastIndexOf('.');
    string type = index > -1 && index < file.FileName.Length - 1 ?
    file.FileName.Substring(index) : ".docx";
    file.CopyTo(stream);
    stream.Position = 0;

    Syncfusion.EJ2.DocumentEditor.WordDocument document =
    Syncfusion.EJ2.DocumentEditor.WordDocument.Load(stream, GetFormatType(type.ToLower()));
    string sfdt = Newtonsoft.Json.JsonConvert.SerializeObject(document);
    document.Dispose();
    return sfdt;
}

internal static Syncfusion.EJ2.DocumentEditor.FormatType GetFormatType(string format)
{

```

```

if (string.IsNullOrEmpty(format))
throw new System.NotSupportedException("EJ2 DocumentEditor does not support this file format.");
switch (format.ToLower()) {
case ".dotx":
case ".docx":
case ".docm":
case ".dotm":
return Syncfusion.EJ2.DocumentEditor.FormatType.Docx;
case ".dot":
case ".doc":
return Syncfusion.EJ2.DocumentEditor.FormatType.Doc;
case ".rtf":
return Syncfusion.EJ2.DocumentEditor.FormatType.Rtf;
case ".txt":
return Syncfusion.EJ2.DocumentEditor.FormatType.Txt;
case ".xml":
return Syncfusion.EJ2.DocumentEditor.FormatType.WordML;
default:
throw new System.NotSupportedException("EJ2 DocumentEditor does not support this file format.");
}
}
`

```

To know about server-side action, please refer this [page](#).

Compatibility with Microsoft Word

Syncfusion Document Editor is a minimal viable Word document viewer/editor product for web applications. As most compatible Word editor, the product vision is adding valuable feature sets of Microsoft Word, and not to cover 100% feature sets of Microsoft Word desktop application. You can even see the feature sets difference between Microsoft Word desktop and their Word online application. So kindly don't misunderstand this component as a complete replacement for Microsoft Word desktop application and expect 100% feature sets of it.

How Syncfusion accepts the feature request for Document Editor

Syncfusion accepts new feature request as valid based on feature value and technological feasibility, then plan to implement unsupported features incrementally in future releases in a phase-by-phase manner.

How to report the problems in Document Editor

You can report the problems with displaying, or editing Word documents in Document Editor component through [support forum](#), [Direct-Trac](#), or [feedback portal](#). Kindly share the Word document for replicating the problem easily in minimal time. If you have confidential data, you can replace it and attach the document.

Why the document pagination differs from Microsoft Word

For your understanding about the Word document structure and the workflow of Word viewer/editor components, the Word document is a flow document in which content will not be preserved page by page; instead, the content will be preserved sequentially like a HTML file. Only the Word viewer/editor paginates the content of the Word document page by page dynamically, when opened for viewing or editing and this page-wise position information will not be preserved in the document level (it is Word file format specification standard). Syncfusion Document Editor component also does the same.

At present there is a known technical limitation related to slight difference in text size calculated using HTML element based text measuring approach. Even though the text size is calculated with correct font and font size values, the difference lies; it is as low as 0.00XX to 0. XXXX values compared to that of Microsoft Word application's display. Hence the document pagination (page-by-page display) can't be guaranteed for all the Word documents to match the pagination of Microsoft Word application.

How Syncfusion address the document pagination difference compared to Microsoft Word

The following table illustrates the reasons for pagination (page-by-page display) difference compared to Microsoft Word in your documents and how Syncfusion address it.

Root causes	How is it solved?
Any mistake (wrong behavior handled) in lay outing the supported elements and formatting	Customer can report to Syncfusion support and track the status through bug report link. Syncfusion fixes the bugs in next possible weekly patch release and service pack or main releases.
Font missing in deployment environment	Customer can either report to Syncfusion support and get suggestion or solve it on their own by installing the missing fonts in their deployment environment.
Any unsupported elements or formatting present in your document	Customer can report to Syncfusion support and track the status through feature request link. Syncfusion implements unsupported features incrementally in future releases based on feature importance, customer interest, efforts involved, and technological feasibility. Also, suggests alternate approach for possible cases.
Technical limitation related to framework	For example, there is a known case with slight fractional difference in text size measured using HTML and Microsoft Word's display. Customer can report to Syncfusion support and track the status through feature request link. Syncfusion does research about alternate approaches to overcome the technical limitation/behaviors and process it same as a feature.
>Note: Here the challenge is, time schedule for implementation varies based on the alternate solution and its reliability.	

See Also

- [Feature modules](#)
- [How to show and hide spinner while opening document in DocumentEditor](#)

Export in ##Platform_Name## Document editor control

Document Editor exports the document into various known file formats in client-side such as Microsoft Word document (.docx), text document (.txt), and its own format called **Syncfusion Document Text (.sfdt)**.

We are providing two types of save APIs as mentioned below.

API name	Purpose
-----	-----
save(filename,FormatType):void	FormatType: Sfdt or Docx or Txt Creates the document with specified file name and format type. Then, the created file is downloaded in the client browser by default.
saveAsBlob(FormatType):Blob	Creates the document in specified format type and returns the created document as Blob. This blob can be uploaded to your required server, database, or file path.

SFDT export

The following example shows how to export documents in Document Editor as Syncfusion document text (.sfdt).

INDEX.JS

```
var documenteditor = new ej.documenteditor.DocumentEditor({
  enableSfdtExport: true, enableSelection: true, enableEditor: true,
  isReadOnly: false });
documenteditor.appendTo('#DocumentEditor');
document.getElementById('export').addEventListener('click', function () {
  documenteditor.save('sample', 'Sfdt');
});
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
documenteditor/styles/fabric.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div style="display: inline">
      <button id="export" class="e-control e-btn e-
primary">Save</button>
    </div>
```

```

        <div style="width:100%;height: 100%">
            <!--Element which will render as DocumentEditor -->
            <div id="DocumentEditor"></div>
        </div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Word export

The following example shows how to export the document as Word document (.docx).

Note: The Syncfusion Document Editor component's document pagination (page-by-page display) can't be guaranteed for all the Word documents to match the pagination of Microsoft Word application. For more information about [why the document pagination \(page-by-page display\) differs from Microsoft Word](#)

INDEX.JS

```

var documenteditor = new ej.documenteditor.DocumentEditor({
enableWordExport: true, enableSelection: true, enableEditor: true,
isReadOnly: false });
documenteditor.appendTo('#DocumentEditor');
document.getElementById('export').addEventListener('click', function () {
    documenteditor.save('sample', 'Docx');
});

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
documenteditor/styles/fabric.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div style="display: inline">

```

```

        <button id="export" class="e-control e-btn e-
primary">Save</button>
    </div>
    <div style="width:100%;height: 100%">
        <!--Element which will render as DocumentEditor -->
        <div id="DocumentEditor"></div>
    </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Text export

The following example shows how to export document as text document (.txt).

INDEX.JS

```

var documenteditor = new ej.documenteditor.DocumentEditor({
enableTextExport: true, enableSelection: true, enableEditor: true,
isReadOnly: false });
documenteditor.appendTo('#DocumentEditor');
document.getElementById('export').addEventListener('click', function () {
    documenteditor.save('sample', 'Txt');
});

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
documenteditor/styles/fabric.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div style="display: inline">
            <button id="export" class="e-control e-btn e-
primary">Save</button>
        </div>
        <div style="width:100%;height: 100%">

```



```

        <!--Element which will render as DocumentEditor -->
        <div id="DocumentEditor"></div>
    </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Export as blob

Document Editor also supports API to store the document into a blob. Refer to the following sample to export document into blob in client-side.

```
`ts
```

```
import { DocumentEditor, FormatType, WordExport, SfdtExport } from '@syncfusion/ej2-
documenteditor';
```

```
//Inject require modules for Export.
```

```
DocumentEditor.Inject(WordExport, SfdtExport);
```

```
let documenteditor: DocumentEditor = new DocumentEditor({ enableSfdtExport: true,
enableWordExport: true, enableTextExport: true });
```

```
documenteditor.appendTo('#DocumentEditor');
```

```
documenteditor.open(sfdt);
```

```
document.getElementById('export').addEventListener('click', () => {
```

```
//Export the current document as Blob object.
```

```
documenteditor.saveAsBlob('Docx').then((exportedDocument: Blob) => {
```

```
// The blob can be processed further
```

```
});
```

```
});
```

```
`
```

For instance, to export the document as Rich Text Format file, implement an ASP.NET MVC web API controller using DocIO library by passing the DOCX blob. Refer to the following code example.

```
`c#
```

```
//API controller for the conversion.
```

```
[HttpPost]
```

```
public HttpResponseMessage ExportAsRtf()
```

```
{
```

```
System.Web.HttpPostedFile data = HttpContext.Current.Request.Files[0];
```

```
//Opens document stream
WordDocument wordDocument = new WordDocument(data.InputStream);
MemoryStream stream = new MemoryStream();
//Converts document stream as RTF
wordDocument.Save(stream, FormatType.Rtf);
wordDocument.Close();
stream.Position = 0;
return new HttpResponseMessage() { Content = new StreamContent(stream) };
}
`
```

In client-side, you can consume this web service and save the document as Rich Text Format (.rtf) file. Refer to the following example.

```
`ts
document.getElementById('export').addEventListener('click', () => {
//Expor the document as Blob object.
documenteditor.saveAsBlob('Docx').then((exportedDocument: Blob) => {
// The blob can be processed further
let formData: FormData = new FormData();
formData.append('fileName', 'sample.docx');
formData.append('data', exportedDocument);
saveAsRtf(formData);
});
});
function saveAsRtf(formData: FormData): void {
//Send the blob object to server to process further.
let httpRequest: XMLHttpRequest = new XMLHttpRequest();
httpRequest.open('POST', '/api/DocumentEditor/ExportAsRtf', true);
httpRequest.onreadystatechange = () => {
if (httpRequest.readyState === 4) {
if (httpRequest.status === 200 || httpRequest.status === 304) {
if (!(!navigator.msSaveBlob)) {
navigator.msSaveBlob(httpRequest.response, 'sample.rtf');
} else {
```

```

let downloadLink: HTMLAnchorElement =
document.createElementNS('http://www.w3.org/1999/xhtml', 'a') as HTMLAnchorElement;
download('sample.rtf', 'rtf', httpRequest.response, downloadLink, 'download' in downloadLink);
}
} else {
console.error(httpRequest.response);
}
}
}
httpRequest.responseType = 'blob';
httpRequest.send(formData);
}
//Download the document in client side.
function download(fileName: string, extension: string, buffer: Blob, downloadLink:
HTMLAnchorElement, hasDownloadAttribute: Boolean): void {
if (hasDownloadAttribute) {
downloadLink.download = fileName;
let dataUrl: string = window.URL.createObjectURL(buffer);
downloadLink.href = dataUrl;
let event: MouseEvent = document.createEvent('MouseEvent');
event.initEvent('click', true, true);
downloadLink.dispatchEvent(event);
setTimeout(() : void => {
window.URL.revokeObjectURL(dataUrl);
dataUrl = undefined;
});
} else {
if (extension !== 'docx' && extension !== 'xlsx') {
let url: string = window.URL.createObjectURL(buffer);
let isPopupBlocked: Window = window.open(url, '_blank');
if (!isPopupBlocked) {
window.location.href = url;
}
}
}
}

```

```
}  
}  
,
```

See Also

- [Feature modules](#)

Server Deployment

Word processor server docker image overview in `##Platform_Name##` Document editor control

The Syncfusion **Word Processor (also known as Document Editor)** is a component with editing capabilities like Microsoft Word. It is used to create, edit, view, and print Word documents. It provides all the common word processing abilities, including editing text; formatting contents; resizing images and tables; finding and replacing text; importing, exporting, and printing Word documents; and using bookmarks and tables of contents.

This Docker image is the predefined Docker container of Syncfusion's Word Processor backend. You can deploy it quickly to your infrastructure.

Word Processor is a commercial product, and it requires a valid license to use it in a production environment ([request license or trial key](#)).

The Word Processor is supported in the JavaScript, Angular, React, Vue, ASP.NET Core, ASP.NET MVC, and Blazor platforms.

Prerequisites

Have [Docker](#) installed in your environment:

- On Windows, install [Docker for Windows](#).
- On macOS, install [Docker for Mac](#).

How to deploy Word Processor Docker image

Step 1: Pull the word-processor-server image from Docker Hub.

```
`console
```

```
docker pull syncfusion/word-processor-server
```

```
,
```

Step 2: Create the docker-compose.yml file with the following code in your file system.

```
`yaml
```

```
version: '3.4'
```

```
services:
```

```
word-processor-server:
```

```
image: syncfusion/word-processor-server:latest
```

```
environment:
```

Provide your license key for activation

SYNCFUSIONLICENSEKEY: YOURLICENSEKEY

ports:

- "6002:80"

Step 3: In a terminal tab, navigate to the directory where you've placed the docker-compose.yml file and execute the following.

```
`console
```

```
docker-compose up
```

Now the Word Processor server Docker instance runs in the localhost with the provided port number <http://localhost:6002>. Open this link in a browser and navigate to the Word Processor Web API control <http://localhost:6002/api/documenteditor>. It returns the default get method response.

```
<!DOCTYPE html>
```

```
<html xmlns="http://www.w3.org/1999/xhtml">
```

```
<head>
```

```
<title>Essential JS 2</title>
```

```
<!-- EJ2 Document Editor dependent material theme -->
```

```
<link href="resources/base/styles/material.css" rel="stylesheet" type="text/css" rel='nofollow' />
```

```
<link href="resources/buttons/styles/material.css" rel="stylesheet" type="text/css" rel='nofollow' />
```

```
<link href="resources/inputs/styles/material.css" rel="stylesheet" type="text/css" rel='nofollow' />
```

```
<link href="resources/popups/styles/material.css" rel="stylesheet" type="text/css" rel='nofollow' />
```

```
<link href="resources/lists/styles/material.css" rel="stylesheet" type="text/css" rel='nofollow' />
```

```
<link href="resources/navigations/styles/material.css" rel="stylesheet" type="text/css" rel='nofollow' />
```

```
<link href="resources/splitbuttons/styles/material.css" rel="stylesheet" type="text/css" rel='nofollow' />
```

```
<link href="resources/dropdowns/styles/material.css" rel="stylesheet" type="text/css" rel='nofollow' />
```

```
<!-- EJ2 DocumentEditor material theme -->
```

```
<link href="resources/documenteditor/styles/material.css" rel="stylesheet" type="text/css"
rel='nofollow' />
```

```
<!-- EJ2 Document Editor dependent scripts -->
```

```
<!-- EJ2 Document Editor script -->
```

```
</head>
```

```
<body>
```

```
<!--element which is going to render-->
<div id='DocumentEditor' style='height:620px'>
</div>
<script>
// Initialize DocumentEditorContainer component.
var documenteditorContainer = new ej.documenteditor.DocumentEditorContainer({ enableToolbar: true
});
//Inject require modules.
ej.documenteditor.DocumentEditorContainer.Inject(ej.documenteditor.Toolbar);
documenteditorContainer.serviceUrl = "http://localhost:6002/api/documenteditor/";
//DocumentEditorContainer control rendering starts
documenteditorContainer.appendTo('#DocumentEditor');
</script>
</body>
</html>
`
```

[How to configure spell checker dictionaries path in Docker compose file](#)

Step 1: In the Docker compose file, mount the local directory as a container volume using the following code.

```
`yaml
version: '3.4'
services:
word-processor-server:
image: syncfusion/word-processor-server:latest
environment:
Provide your license key for activation
SYNCFUSIONLICENSEKEY: YOURLICENSEKEY
volumes:
  • ./data:/app/data

ports:
  • "6002:80"
`
```

This YAML definition binds the data folder that is available in the Docker compose file directory.

Step 2: In the data folder, include the dictionary files (.dic, .aff) and JSON file. The JSON file should contain the language based dictionary file configuration in the following format.

```
`yaml
[
{
  "LanguageID": 1036,
  "DictionaryPath": "fr_FR.dic",
  "AffixPath": "fr_FR.aff",
  "PersonalDictPath": "customDict.dic"
},
{
  "LanguageID": 1033,
  "DictionaryPath": "en_US.dic",
  "AffixPath": "en_US.aff",
  "PersonalDictPath": "customDict.dic"
}
]
```

Note: By default, the json file name should be "spellcheck.json". You can also use different file name by mounting the file name to 'SPELLCHECKJSONFILENAME' attribute in Docker compose file as below,

```
`yaml
version: '3.4'
services:
  word-processor-server:
    image: syncfusion/word-processor-server:latest
    environment:
      Provide your license key for activation
      SYNCFUSIONLICENSEKEY: YOURLICENSEKEY
      SPELLCHECKDICTIONARYPATH: data
      SPELLCHECKJSONFILENAME: spellcheck1.json
    volumes:
      • ./data:/app/data
    ports:
```

- "6002:80"

,

Step 3: For handling the personal dictionary, place an empty .dic file (e.g., customDict.dic file) in the data folder.

Step 4: Provide the configured volume path to the environment variable like in the following in the Docker compose file.

```
`yaml
```

```
version: '3.4'
```

```
services:
```

```
word-processor-server:
```

```
image: syncfusion/word-processo -server:latest
```

```
environment:
```

```
Provide your license key for activation
```

```
SYNCFUSIONLICENSEKEY: YOURLICENSEKEY
```

```
SPELLCHECKDICTIONARYPATH: data
```

```
volumes:
```

- ./data:/app/data

```
ports:
```

- "6002:80"

,

How to copy template Word documents to Docker image

You can copy the required template Word documents into docker container while deploying the docker image to server. You can open these Word documents present in the server by passing the document path (name with relative path) to LoadDocument() web API.

Note: Place the word files in the data folder mentioned in the volumes section(i.e., C:/Docker/Data) of the docker-compose.yml file. All the files present in the folder path (C:/Docker/Data) mentioned in the volumes section of 'docker-compose.yml' file will be copied to the respective folder (/app/Data) of docker container. The Word documents copied to docker container can be processed using the 'LoadDocument' web API.

The following code example shows how to use LoadDocument() API in Document Editor.

```
`ts
```

```
import { DocumentEditorContainer, Toolbar } from '@syncfusion/ej2-documenteditor';
```

```
let container: DocumentEditorContainer = new DocumentEditorContainer({ enableToolbar: true, height: '590px' });
```



```

DocumentEditorContainer.Inject(Toolbar);
container.created = function () {
var dataContext = this;
var uploadDocument = new FormData();
uploadDocument.append('DocumentName', 'Getting Started.docx');
var baseUrl = 'http://localhost:6002/api/documenteditor/LoadDocument';
var httpRequest = new XMLHttpRequest();
httpRequest.open('POST', baseUrl, true);
httpRequest.onreadystatechange = function () {
if (httpRequest.readyState === 4) {
if (httpRequest.status === 200 || httpRequest.status === 304) {
dataContext.container.documentEditor.open(httpRequest.responseText);
}
}
};
httpRequest.send(uploadDocument);
dataContext.container.documentEditor.spellChecker.languageID = 1033;
};
container.appendTo('#container');
`

```

Refer to these getting started pages to create a Word Processor in [Angular](#), [React](#), [Vue](#), [ASP.NET MVC](#), [ASP.NET Core](#), and [Blazor](#).

How to deploy word processor server docker container in azure app service in
 ##Platform_Name## Document editor control

Prerequisites

- Have [Azure account](#) and [Azure CLI](#) setup in your environment.
- Run the following command to open the Azure login page. Sign into your [Microsoft Azure account](#).

`

az login

`

Step 1: Create a resource group.

Create a resource group using the [az group create](#) command.

The following example creates a resource group named `documenteditorresourcegroup` in the `eastus` location.

,

```
az group create --name documenteditorresourcegroup --location "East US"
```

,

Step 2: Create an Azure App Service plan.

Create an App Service plan in the resource group with the [az appservice plan create](#) command.

The following example creates an App Service plan named `documenteditorappservice` in the Standard pricing tier (`--sku S1`) and in a Linux container (`--is-linux`).

,

```
az appservice plan create --name documenteditorappservice --resource-group
documenteditorresourcegroup --sku S1 --is-linux
```

,

Step 3: Create a Docker Compose app.

Create a multi-container [web app](#) in the `documenteditorappservice` App Service plan with the [az webapp create](#) command. The following command creates the web app using the provided Docker compose file. Please look into the section for getting started with Docker compose to create the Docker compose file for the Document Editor server and use the created Docker compose file here.

,

```
az webapp create --resource-group documenteditorresourcegroup --plan documenteditorappservice --
name documenteditor-server --multicontainer-config-type compose --multicontainer-config-file
documenteditor-server-compose.yml
```

,

Step 4: Browse to the app.

Browse to the deployed app at `http://<app_name>.azurewebsites.net`, i.e. `http://documenteditor-server.azurewebsites.net`. Browse this link and navigate to the Document Editor Web API control `http://documenteditor-server.azurewebsites.net/api/documenteditor`. It returns the default get method response.

For more information about the app container service, please look deeper into the [Microsoft Azure Container Service](#) for a production-ready setup.

How to deploy word processor server docker container in azure kubernetes service in
##Platform_Name## Document editor control

Prerequisites

- Have [Azure account](#) and [Azure CLI](#) setup in your environment.
- Run the following command to open the Azure login page. Sign into your [Microsoft Azure account](#).

,

```
az login
```

```
,
```

Step 1: Create a resource group.

Create a resource group using the [az group create](#) command.

The following example creates a resource group named `documenteditorresourcegroup` in the `eastus` location.

```
,
```

```
az group create --name documenteditorresourcegroup --location "East US"
```

```
,
```

Step 2: Create AKS cluster.

Use the [az aks create](#) command to create an AKS cluster. The following example creates a cluster named `documenteditorcluster` with one node.

```
,
```

```
az aks create --resource-group documenteditorresourcegroup --name documenteditorcluster --node-count 1
```

```
,
```

Step 3: Connect to the cluster.

Install the [kubectl](#) into the workspace using the following command.

```
,
```

```
az aks install-cli
```

```
,
```

To configure `kubectl` to connect to your Kubernetes cluster, use the [az aks get-credentials](#) command. This command downloads credentials and configures the Kubernetes CLI to use them.

```
,
```

```
az aks get-credentials --resource-group documenteditorresourcegroup --name documenteditorcluster
```

```
,
```

Step 4: Create Kubernetes Services and Deployments

[Kubernetes Services](#) and [Deployments](#) can be configured in a file. To run the Document Editor server, you must define a Service and a Deployment `documenteditorserver`. To do this, create the `documenteditor-server.yml` file in the current directory using the following code.

```
`yaml
```

```
apiVersion: apps/v1
```

```
kind: Deployment
```

```
metadata:
```

```
labels:
```

```
app: documenteditorserver
name: documenteditorserver
spec:
  replicas: 1
  selector:
    matchLabels:
      app: documenteditorserver
  strategy: {}
  template:
    metadata:
      labels:
        app: documenteditorserver
    spec:
      containers:
        - image: syncfusion/word-processor-server:latest

name: documenteditorserver
ports:
  - containerPort: 80

env:
  - name: SYNCFUSIONLICENSEKEY
    value: "YOURLICENSEKEY"
  apiVersion: v1
  kind: Service
  metadata:
    labels:
      app: documenteditorserver
      name: documenteditorserver
  spec:
    ports:
      - port: 80
```

```
targetPort: 80
selector:
app: documenteditorserver
type: LoadBalancer
`
```

Step 5: To create all Services and Deployments needed to run the Document Editor server, execute the following.

```
`console
kubectl create -f ./documenteditor-server.yml
`
```

Run the following command to get the Kubernetes cluster deployed service details and copy the external IP address of the Document Editor service.

```
`console
kubectl get all
`
```

Browse the copied external IP address and navigate to the Document Editor Web API control `http://<external-ip>/api/documenteditor`. It returns the default get method response.

For more details about the Azure Kubernetes service, please look deeper into [Microsoft Azure Kubernetes Service](#) for a production-ready setup.

How to publish documenteditor web api application in azure app service from visual studio in `##Platform_Name##` Document editor control

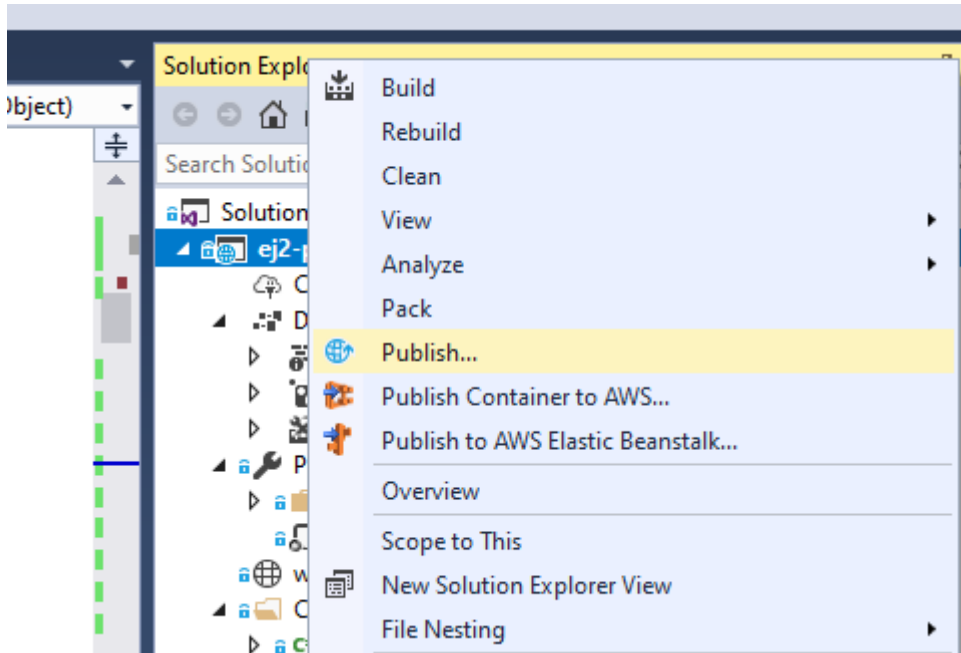
Prerequisites

- Visual Studio 2017 or 2019.
- An [Azure subscription](#).
- The Document Editor Web API controller application from [here](#).

Make sure you build the project using the Build > Build Solution menu command before following the deployment steps.

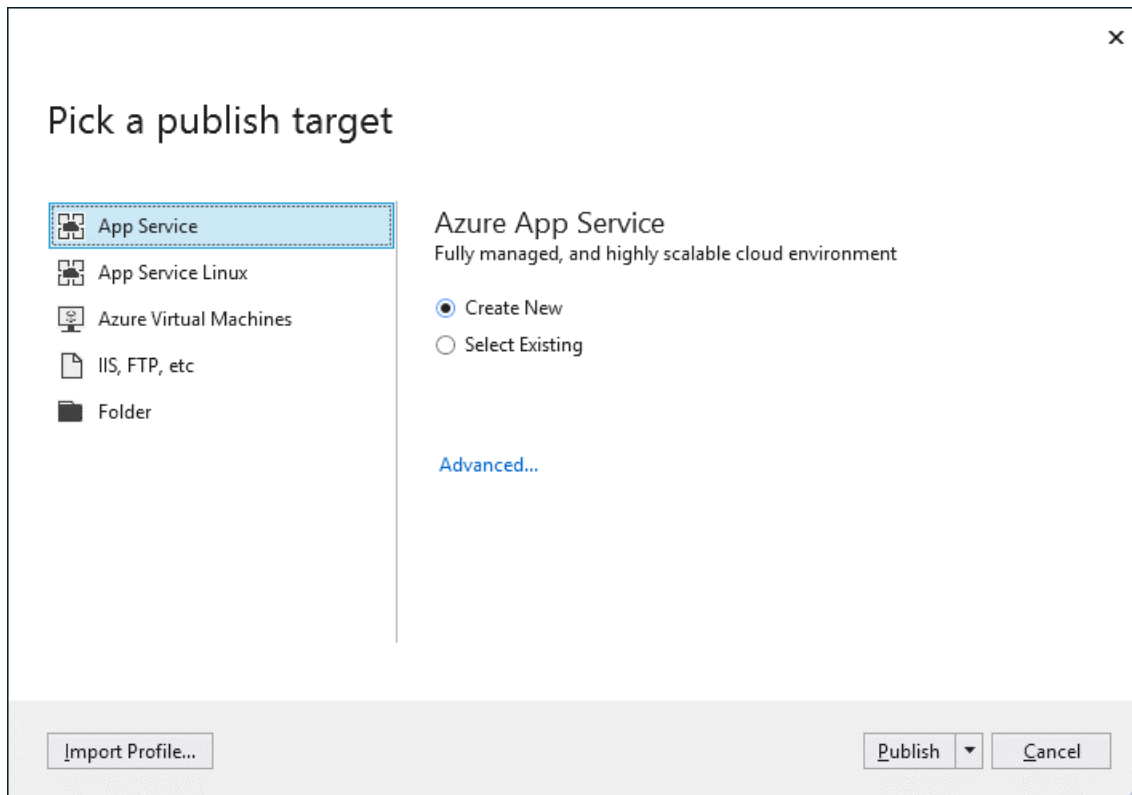
Publish to Azure App Service

Step 1: In Solution Explorer, right-click the project and click Publish (or use the Build > Publish menu item).



Step 2: If you have previously configured any publishing profiles, the Publish pane appears, in which case select Create new profile.

Step 3: In the Pick a publish target dialog box, select App Service.



Step 4: Select Publish. The Create App Service dialog box appears. Sign in with your Azure account, if necessary, and then the default app service settings populate the fields.

App Name
ej2-documenteditor-server20200514102909

Subscription
Microsoft Azure Enterprise

Resource Group
cloud-shell-storage-centralindia (centralindia) [New...](#)

Hosting Plan
ej2-documenteditor-server20200514102909P* (South Centra [New...](#)

Application Insights
None

Explore additional Azure services

- [Create a SQL Database](#)
- [Create a storage account](#)

Clicking the Create button will create the following Azure resources

- Hosting Plan - ej2-documenteditor-server202005141...
- App Service - ej2-documenteditor-server20200514102909

[Export...](#) [Create](#) [Cancel](#)

Step 5: Select Create. Visual Studio deploys the app to your Azure App Service, and the web app loads in your browser with the app name at http://<app_name>.azurewebsites.net (i.e. <http://ej2-documenteditor-server20200514102909.azurewebsites.net>).

Step 6: Navigate to Document Editor Web API control <http://ej2-documenteditor-server20200514102909.azurewebsites.net/api/documenteditor>. It returns the default get method response.

For more information about the app container service, please look deeper into the [Microsoft Azure App Service](#) for a production-ready setup.

How to deploy documenteditor java web api in azure in ##Platform_Name## Document editor control

Prerequisites

Have [Azure account](#) and [Azure CLI](#) setup in your environment.

You can get the example [web service project from GitHub](#) and then perform the following steps to create the packages and host in azure app service.

Step 1: Clean the package using following command.

```
`console
mvn clean package
`
```

Step 2: Run the application locally using following command.

```
`console
```

```
mvn spring-boot:run
```

```
,
```

Step 3: Build the package using following command.

```
`console
```

```
mvn package
```

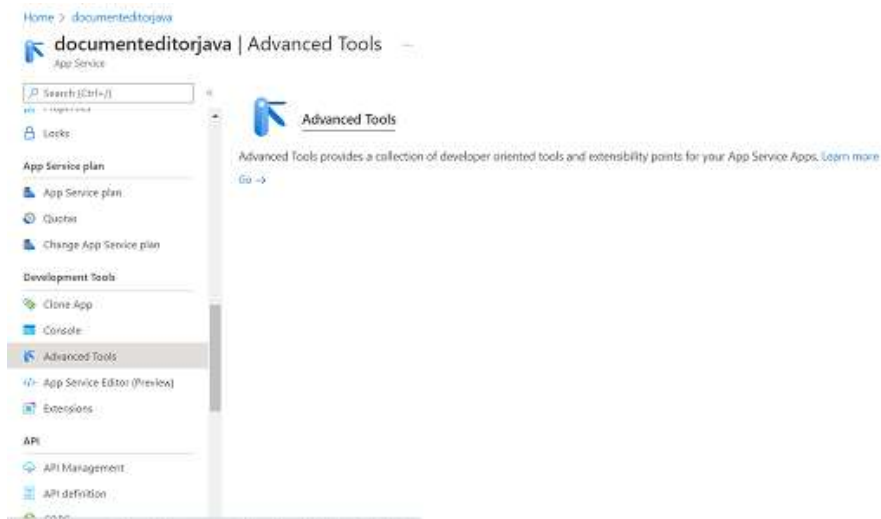
```
,
```

Above package generation command creates the **tomcat-0.0.1-SNAPSHOT.war** in the below location in the sample folder.

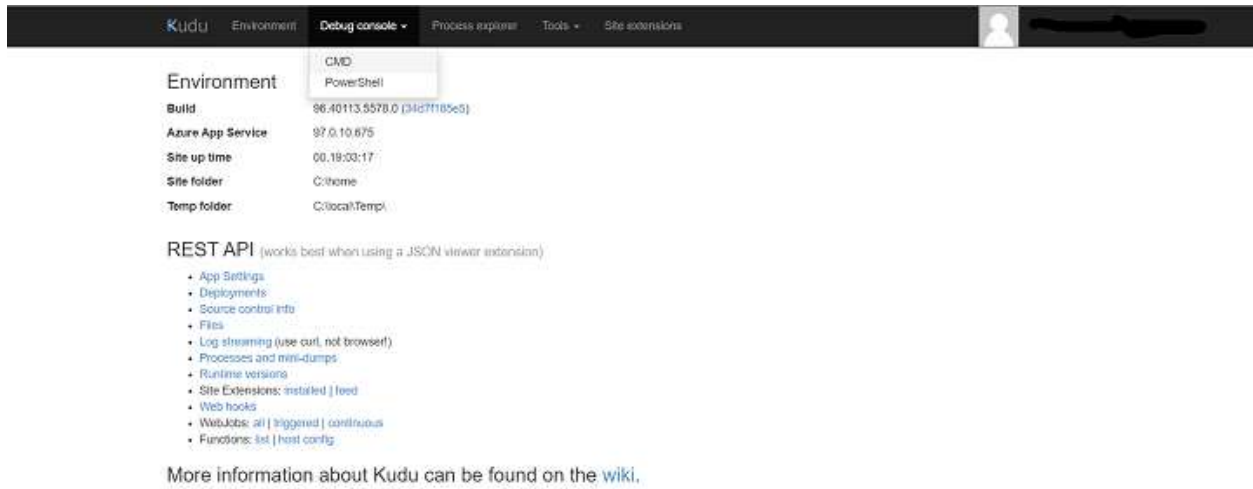
target/tomcat-0.0.1-SNAPSHOT.war

Step 4: Create a Azure app service with Java & Tomcat. For example, create the app services name as **documenteditorjava**.

Step 5: After creating app service, navigate to **Advanced Tools** options under **Development Tools**.



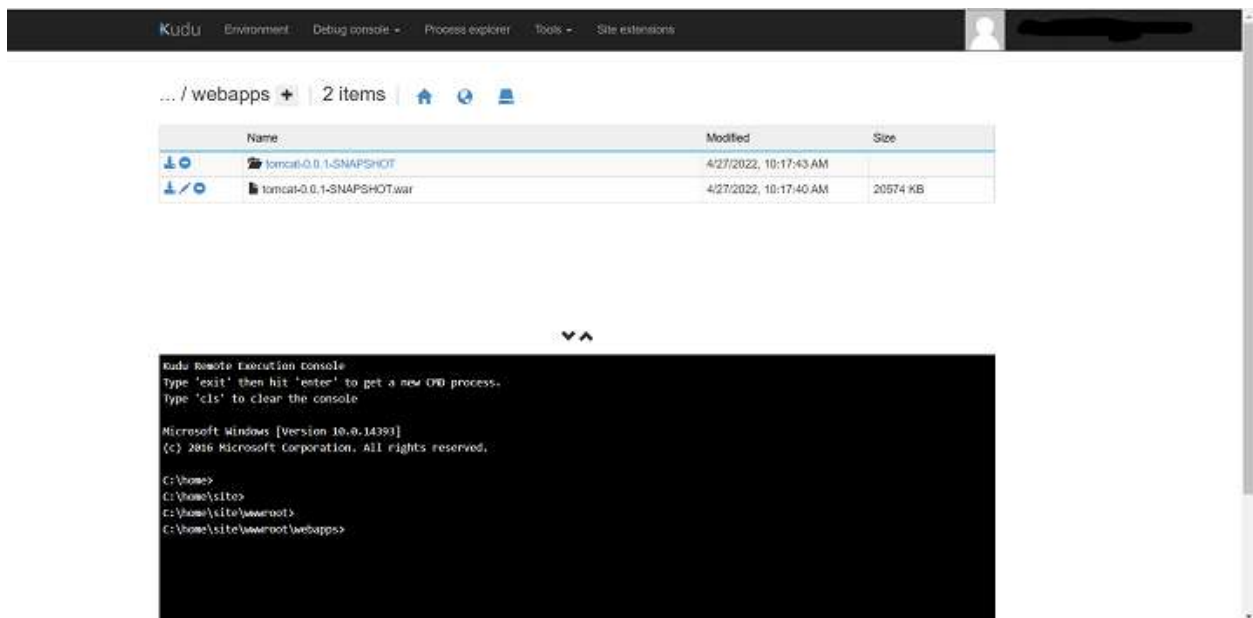
Then, click **Go** and select the **CMD** options under **Debug console**.



Step 6: Once the file manager is opened, please navigate to

site -> wwwroot -> webapps

Step 7: Now, upload the generated war file `tomcat-0.0.1-SNAPSHOT.war`. Uploaded war file gets extracted automatically, it will be uploaded like below:



Step 8: Browse to the app.

Browse to the deployed app at `http://<app_name>.azurewebsites.net`, i.e. `http://documenteditorjava.azurewebsites.net`. Browse this link and it navigates to the Document Editor Web API control `http://documenteditorjava.azurewebsites.net/tomcat-0.0.1-SNAPSHOT`. It returns the default get method response.

Accessibility in Angular Document editor component

The accessibility compliance for the Document editor component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2 Support](#) | |

| [Section 508 Support](#) | |

| [Screen Reader Support](#) | |

| [Right-To-Left Support](#) | |

| [Color Contrast](#) | |

| [Mobile Device Support](#) | |

| [Keyboard Navigation Support](#) | |

| [Accessibility Checker Validation](#) | |

| [Axe-core Accessibility Validation](#) | |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<div> - All features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>

Keyboard interaction

Document editor supports [keyboard shortcuts](#).

Ensuring accessibility

The Document editor component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Document editor component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Document editor component with accessibility tools.

See also

- [Accessibility in Syncfusion Angular components](#)

Image in ##Platform_Name## Document editor control

Document Editor supports common raster format images like PNG, BMP, JPEG, SVG and GIF. You can insert an image file or online image in the document using the [insertImage\(\)](#) method. Refer to the following sample code.

INDEX.JS

```
var documenteditor = new ej.documenteditor.DocumentEditor({
  isReadOnly: false,
  enableEditor: true,
  enableSection: true,
  enableImageResizer: true,
  enableEditorHistory: true
});
documenteditor.appendTo('#DocumentEditor');
//Insert Image From URL with alternate text
documenteditor.editor.insertImage('https://cdn.syncfusion.com/content/images/Logo/Logo_Black_72dpi_without.png', 200, 200, 'Syncfusion');
document.getElementById('insert-picture').addEventListener('click', () => {
  var pictureUpload = document.getElementById("insertImageButton");
  pictureUpload.value = '';
  pictureUpload.click();
});
document.getElementById('insertImageButton').addEventListener('change',
onInsertImage);
function onInsertImage(args) {
  if (navigator.userAgent.match('Chrome') ||
  navigator.userAgent.match('Firefox') || navigator.userAgent.match('Edge') ||
  navigator.userAgent.match('MSIE') || navigator.userAgent.match('.NET')) {
    if (args.target.files[0]) {
      var path = args.target.files[0];
      var reader = new FileReader();
      reader.onload = function (frEvent) {
        var base64String = frEvent.target.result;
        var image = document.createElement('img');
        image.addEventListener('load', function () {
          documenteditor.editor.insertImage(base64String,
this.width, this.height);
        })
        image.src = base64String;
      };
      reader.readAsDataURL(path);
    }
    //Safari does not Support FileReader Class
  } else {
    var image = document.createElement('img');
    image.addEventListener('load', function () {
```

```

        documenteditor.editor.insertImage(args.target.value);
    })
    image.src = args.target.value;
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
documenteditor/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/fabric.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <input type="file" id="insertImageButton" style="position:fixed;
left:-110em" accept=".jpg,.jpeg,.png,.bmp">
    <div id="toolbar">
      <button id="insert-picture">Image</button>
    </div>
    <div style="width:100%;height: 100%">
      <!--Element which will render as DocumentEditor -->
      <div id="DocumentEditor"></div>
    </div>
  </div>
</body>
<script>
var ele = document.getElementById('container');
if(ele) {

```

```

    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Note: 1. Image files will be internally converted to base64 string. Whereas, online images are preserved as URL. N> 2. EMF and WMF images can't be inserted, but these types of images will be preserved in Document Editor when using ASP.NET MVC Web API.

Alternate text

Document Editor expose API to get or set the alternate text of the selected image. Refer to the following sample code.

```
`ts
```

```
documenteditor.selection.imageFormat.alternateText = 'Adventure Cycle';
```

```
,
```

Image resizing

Document Editor provides built-in image resizer that can be injected into your application based on the requirements. This allows you to resize the image by dragging the resizing points using mouse or touch interactions. This resizer appears as follows.



Changing size

Document Editor expose API to get or set the size of the selected image. Refer to the following sample code.

```
`ts
```

```
documenteditor.selection.imageFormat.width = 800;
```

```
documenteditor.selection.imageFormat.height = 800;
```

```
,
```

Note: Images are stored and processed(read/write) as base64 string in DocumentEditor. The online image URL is preserved as a URL in DocumentEditor upon saving.

Text wrapping style

Text wrapping refers to how images fit with surrounding text in a document. Please [refer to this page](#) for more information about text wrapping styles available in Word documents.

Positioning the image

DocumentEditor preserves the position properties of the image and displays the image based on position properties. It does not support modifying the position properties. Whereas the image will be automatically moved along with text edited if it is positioned relative to the line or paragraph.

See Also

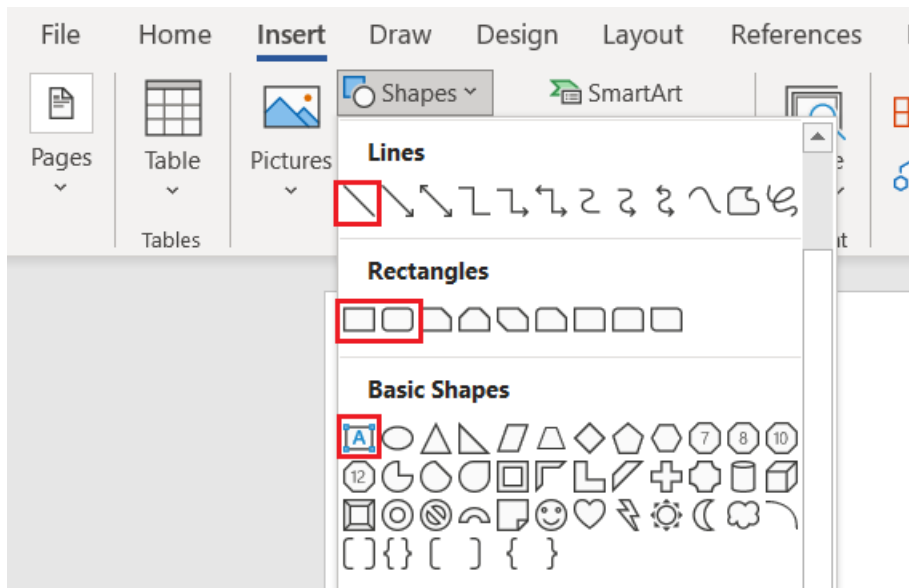
- [Feature modules](#)

Shapes in ##Platform_Name## Document editor control

Shapes are drawing objects that include a text box, rectangles, lines, curves, circles, etc. It can be preset or custom geometry. At present, DocumentEditor does not have support to insert shapes. however, if the document contains a shape while importing, it will be preserved properly.

Supported shapes


The DocumentEditor has preservation support for Text box, Rectangle, Rounded Rectangle and Line shapes.



Note: When using ASP.NET MVC service, the unsupported shapes will be converted as image and preserved as image.

Text box Shape

A text box is a rectangular area on the document where you can enter text. When you click in a text box, a flashing cursor will display indicating that you can begin typing. It allows you to enter multiple lines of text with all text formatting.

Adventure Works Cycles, the fictitious company on which the Adventure Works sample databases are based, is a large, multinational manufacturing company. The company manufactures and sells metal  and composite bicycles to North American, European and Asian commercial markets. While its base operation is located in Bothell, Washington with 290 employees, several regional sales teams are located throughout their market base.

Shape Resizer

The DocumentEditor also supports a built-in shape resizer to resize the shapes present in the document. The shape resizer accepts both touch and mouse interactions.



Text wrapping style

Text wrapping refers to how shapes fit with surrounding text in a document. Please [refer to this page](#) for more information about text wrapping styles available in Word documents.

Positioning the shape

DocumentEditor preserves the position properties of the shape and displays the shape based on position properties. It does not support modifying the position properties. Whereas the shape will be automatically moved along with text edited if it is positioned relative to the line or paragraph.

Text wrapping style in ##Platform_Name## Document editor control

Text wrapping refers to how images and shapes are fit with surrounding text in a document. Currently, DocumentEditor has only preservation support for image and textbox shape with below wrapping styles.

In-Line with Text

In this option, the image or shape is placed on the same line surrounding with text like any other word or letter. This image or shape will be automatically moved along with the text while editing, whereas the other options denote that the image or shape stays in a fixed position while the text shifts and wraps around it.

Adventure Works Cycles, the fictitious company on which the AdventureWorks sample databases are based, is a large, multinational manufacturing company. The company



manufactures and sells metal and composite bicycles to North American, European and Asian commercial markets. While its base operation is located in

In Front of Text

In this option, the image or shape is placed in front of the text. This can be used to place an image around some text or to add shape to highlight the part in a paragraph.

Adventure Works Cycles, the fictitious company on which the AdventureWorks sample databases are based, is a large manufacturing company. The company manufactures and sells metal and composite bicycles to North American, European and Asian commercial markets. While its base operation is located in Bothell, Washington with 290 employees, several regional sales teams and manufacturer and located and shipped in throughout their market base.



Note: Starting from v18.2.0.x, the in front of wrapping styles are supported.

Top and Bottom

In this option, Text wraps above and below the image or shape. No text is to the left or right of the image or shape. This can be used for larger images or shapes that occupy most of the width in a document.

Note: Starting from v19.1.0.x, the top and bottom wrapping style is supported.

Adventure Works Cycles, the fictitious company on which the AdventureWorks sample databases are based, is a large, multinational manufacturing company. The company



manufactures and sells metal and composite bicycles to North American, European and Asian commercial markets. While its base operation is located in Bothell, Washington with 290

Behind

In this option, the image or shape is placed behind the text. This can be used when you need to add a watermark or background image to a document.

Adventure Works Cycles, the fictitious company on which the AdventureWorks sample databases are based, is a large, multinational manufacturing company. The company manufactures and sells metal and composite bicycles to North American, European and Asian commercial markets. While its base operation is located in Bothell, Washington with 290 employees, several regional sales teams and manufacturer and located and shipped in throughout their market base.



Note: Starting from v19.2.0.x, behind text wrapping styles are supported.

Square

In this option, Text wraps around the image or text box in a square shape.

Note: Tight and Through styles will be preserved as square wrapping style in DocumentEditor which is supported from v19.2.0.x.

Adventure Works Cycles, the fictitious company on which the Adventure Works sample databases are based, is a large, multinational manufacturing company. The company manufactures and sells metal North American, European and While its base operation is with 290 employees, several regional sales teams are located throughout their market base.

Adventure works cycles company.

Bookmark in ##Platform_Name## Document editor control

Bookmark is a powerful tool that helps you to mark a place in the document to find again easily. You can enter many bookmarks in the document and give each one a unique name to identify easily.

Document Editor provides built-in dialog to add, delete, and navigate bookmarks within the document. To add a bookmark, select a portion of text in the document. After that, jump to the location or add links to it within the document using built-in hyperlink dialog. You can also delete bookmarks from a document.

Bookmark names need to begin with a letter. They can include both numbers and letters, but not spaces. To separate the words, use an underscore.

Bookmark names starting with an underscore are called hidden bookmarks. For example, bookmarks generated for table of contents.

Add bookmark

Using [insertBookmark](#) method, Bookmark can be added to the selected text.

```
`c#
```

```
container.documentEditor.editor.insertBookmark("Bookmark1");
```

```
,
```

Select Bookmark

You can select the bookmark in the document using [selectBookmark](#) method by providing Bookmark name to select as shown in the following code snippet.

```
`c#
```

```
container.documentEditor.selection.selectBookmark("Bookmark1", true);
```

```
,
```

Note: Second parameter is optional parameter and it denotes is exclude bookmark start and end from selection. If true, excludes bookmark start and end from selection.

Delete Bookmark

You can delete bookmark in the document using [deleteBookmark](#) method as shown in the following code snippet.

```
`c#
```

```
container.documentEditor.editor.deleteBookmark("Bookmark1");
```

```
,
```

Get Bookmark from document

You can get all the bookmarks in the document using [getBookmarks](#) method as shown in the following code snippet.

```
`c#
container.documentEditor.getBookmarks(false);
`
```

Note: Parameter denotes is include hidden bookmarks. If false, ignore hidden bookmark.

Get Bookmark from selection

You can get bookmarks in current selection in the document using [getBookmarks](#) method as shown in the following code snippet.

```
`c#
container.documentEditor.selection.getBookmarks(false);
`
```

Replace bookmark content

You can replace bookmark content without removing the bookmark start and end for backtracking the bookmark content.

```
`c#
container.documentEditor.selection.selectBookmark("Bookmark1", true);
container.documentEditor.editor.insertText('Hello World')
`
```

You can replace content by removing the bookmark start and end, thus the bookmark content can't be tracked in future.

```
`c#
container.documentEditor.selection.selectBookmark("Bookmark1");
container.documentEditor.editor.insertText('Hello World')
`
```

Show or Hide bookmark

You can show or hide the show square brackets around bookmarked items in Document editor component.

The following example code illustrates how to show or hide square brackets around bookmarked items.

```
`ts
container.documentEditorSettings.showBookmarks = true;
`
```

Bookmark Dialog

The following example shows how to open bookmark dialog in Document Editor.

INDEX.JS

```

var documenteditor = new ej.documenteditor.DocumentEditor({
enableBookmarkDialog: true, enableSelection: true, enableEditor: true,
isReadOnly: false, enableEditorHistory: true });
documenteditor.appendTo('#DocumentEditor');
document.getElementById('dialog').addEventListener('click', function () {
    documenteditor.showDialog('Bookmark');
});

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
documenteditor/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/fabric.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="toolbar">
            <button id="dialog">Dialog</button>
        </div>
        <div style="width:100%;height: 100%">
            <!--Element which will render as DocumentEditor -->
            <div id="DocumentEditor"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {

```

```

    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Feature modules](#)
- [Bookmark dialog](#)

Link in ##Platform_Name## Document editor control

Document Editor supports hyperlink field. You can link a part of the document content to Internet or file location, mail address, or any text within the document.

Navigate a hyperlink

Document Editor triggers 'requestNavigate' event whenever user clicks Ctrl key or tap a hyperlink within the document. This event provides necessary details about link type, navigation URL, and local URL (if any) as arguments, and allows you to easily customize the hyperlink navigation functionality.

Add the requestNavigate event for DocumentEditor

The following example illustrates how to add requestNavigate event for DocumentEditor.

INDEX.JS

```

var documenteditor = new ej.documenteditor.DocumentEditor({ enableSelection:
true });
documenteditor.appendTo('#DocumentEditor');
documenteditor.requestNavigate = (args) => {
    if (args.linkType !== 'Bookmark') {
        var link = args.navigationLink;
        if (args.localReference.length > 0) {
            link += '#' + args.localReference;
        }
        window.open(link);
        args.isHandled = true;
    }
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
documenteditor/styles/fabric.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>

```

```
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="DocumentEditor">

            </div>
        </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Add the requestNavigate event for DocumentEditorContainer component

The following example illustrates how to add requestNavigate event for DocumentEditorContainer component.

`ts

```
import { DocumentEditor, SfddExport, Selection, RequestNavigateEventArgs } from '@syncfusion/ej2-
documenteditor';
```

```
let hostUrl: string =
```

```
'https://ej2services.syncfusion.com/production/web-services/';
```

```
let container: DocumentEditorContainer = new DocumentEditorContainer({
```

```
enableToolbar: true,
```

```
height: '590px',
```

```
});
```

```
DocumentEditorContainer.Inject(Toolbar);
```

```
container.serviceUrl = hostUrl + 'api/documenteditor/';
```

```
container.appendTo('#container');
```

```
// Add event listener for requestNavigate event to customize hyperlink navigation functionality
```

```
container.documentEditor.requestNavigate = (args: RequestNavigateEventArgs) => {
```

```
if (args.linkType !== 'Bookmark') {
```

```
let link: string = args.navigationLink;
```

```
if (args.localReference.length > 0) {
```

```
link += '#' + args.localReference;
```

```
}
```

```
//Navigate to the selected URL.
```

```

window.open(link);
args.isHandled = true;
}
};
`

```

If the selection is in hyperlink, trigger this event by calling 'navigateHyperlink' method of 'Selection' instance. Refer to the following example.

```

`ts
documenteditor.selection.navigateHyperlink();
`

```

Copy link

Document Editor copies link text of a hyperlink field to the clipboard if the selection is in hyperlink. Refer to the following example.

```

`ts
documenteditor .selection.copyHyperlink();
`

```

Add hyperlink

To create a basic hyperlink in the document, press **ENTER** / **SPACEBAR** / **SHIFT + ENTER** / **TAB** key after typing the address, for instance <http://www.google.com>. Document Editor automatically converts this address to a hyperlink field. The text can be considered as a valid URL if it starts with any of the following.

```

`http://`<br>
`https://`<br>
file:///<br>
www.<br>
mailto:<br>

```

Refer to the following example.

INDEX.JS

```

var documenteditor = new ej.documenteditor.DocumentEditor({ enableSelection:
true, isReadOnly: false, enableEditor: true });
documenteditor.appendTo('#DocumentEditor');
documenteditor.requestNavigate = (args) => {
  if (args.linkType !== 'Bookmark') {
    var link = args.navigationLink;
    if (args.localReference.length > 0) {
      link += '#' + args.localReference;
    }
    window.open(link);
    args.isHandled = true;
  }
}

```

```
}
};
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
documenteditor/styles/fabric.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="DocumentEditor">

      </div>
    </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Also Document Editor expose API [insertHyperlink\(\)](#) to insert hyperlink.

Refer to the following sample code.

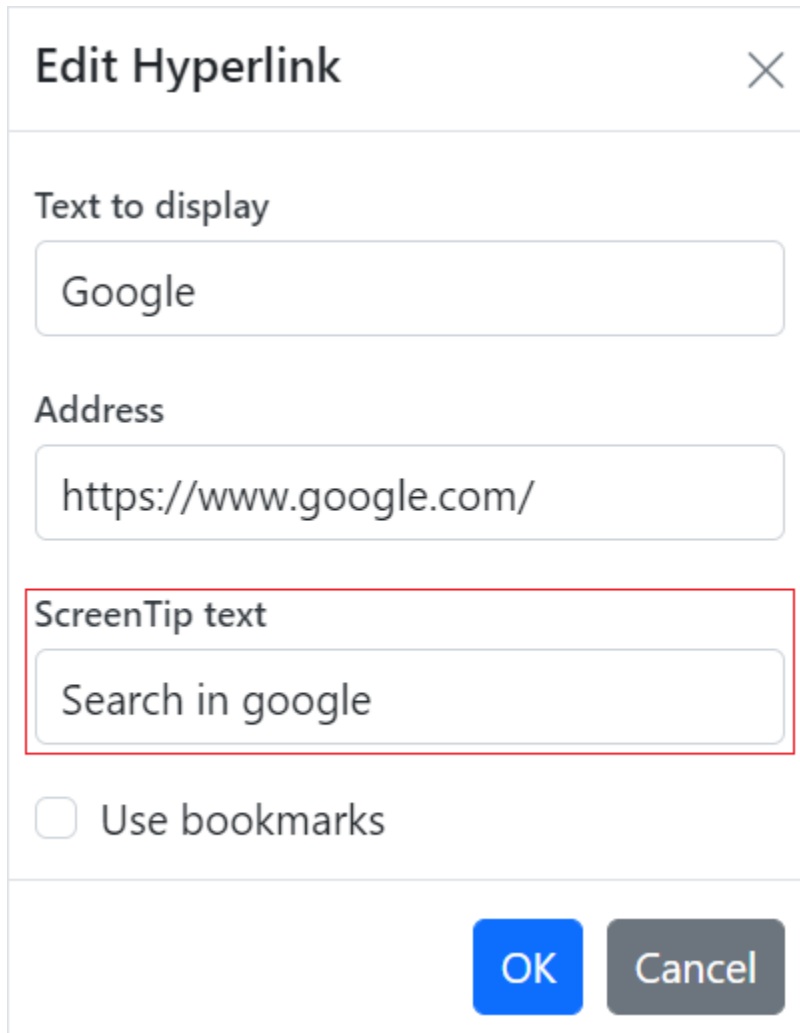
```
`ts
documenteditor.insertHyperlink('https://www.google.com', 'Google');
`
```

Customize screen tip

You can customize the screen tip text for the hyperlink by using below sample code.

```
`ts
documenteditor.insertHyperlink('https://www.google.com', 'Google', '<<Screen tip text>>');
`
```

Screen tip text can be modified through UI by using the [Hyperlink dialog](#)



Remove hyperlink

To remove link from hyperlink in the document, press Backspace key at the end of a hyperlink. By removing the link, it will be converted as plain text. You can use 'removeHyperlink' method of 'Editor' instance if the selection is in hyperlink. Refer to the following example.

`ts

```
documenteditor.editor.removeHyperlink();
```

,

Hyperlink dialog

Document Editor provides dialog support to insert or edit a hyperlink. Refer to the following example.

INDEX.JS

```
var documenteditor = new ej.documenteditor.DocumentEditor({
  enableHyperlinkDialog: true, enableSelection: true, enableEditor: true,
  isReadOnly: false, enableEditorHistory: true });
documenteditor.appendTo('#DocumentEditor');
document.getElementById('dialog').addEventListener('click', function () {
  documenteditor.showDialog('Hyperlink');
});
```


INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
documenteditor/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/fabric.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="toolbar">
      <button id="dialog">Dialog</button>
    </div>
    <div style="width:100%;height: 100%">
      <!--Element which will render as DocumentEditor -->
      <div id="DocumentEditor"></div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

You can use the following keyboard shortcut to open the hyperlink dialog if the selection is in hyperlink.

Key Combination	Description
----- -----	
Ctrl + K	Open hyperlink dialog that allows you to create or edit hyperlink

See Also

- [Feature modules](#)
- [Hyperlink dialog](#)

Table in `##Platform_Name##` Document editor control

Tables are an efficient way to present information. Document Editor can display and edit the tables. You can select and edit tables through keyboard, mouse, or touch interactions. Document Editor exposes a rich set of APIs to perform these operations programmatically.

Create a table

You can create and insert a table at cursor position by specifying the required number of rows and columns.

Refer to the following sample code.

```
`ts
documentedior.editor.insertTable(3,3);
`
```

The maximum size of row and column is limited to 32767 and 63 respectively.

Insert rows

You can add a row (or several rows) above or below the row at cursor position by using the [insertRow](#) method. This method accepts the following parameters:

Parameter	Type	Description
left(optional)	boolean	This is optional and if omitted, it takes the value as false and inserts to the right of column at cursor position.
count(optional)	number	This is optional and if omitted, it takes the value as 1.

Refer to the following sample code.

```
`ts
//Insert a column to the right of the column at cursor position.
documentedior.editor.insertColumn();
//Insert a column to the left of the column at cursor position.
documentedior.editor.insertColumn(false);
//Insert two columns to the left of the column at cursor position.
documentedior.editor.insertColumn(false, 2);
`
```

Select an entire table

If the cursor position is inside a table, you can select the entire table by using the following sample code.

```
`ts
documenteditor.selection.selectTable();
`
```

Select row

You can select the entire row at cursor position by using the following sample code.

```
`ts
documenteditor.selection.selectRow();
`
```

If current selection spans across cells of different rows, all these rows will be selected.

Select column

You can select the entire column at cursor position by using the following sample code.

```
`ts
documenteditor.selection.selectColumn();
`
```

If current selection spans across cells of different columns, all these columns will be selected.

Select cell

You can select the cell at cursor position by using the following sample code.

```
`ts
documenteditor.selection.selectCell();
`
```

Delete table

Document Editor allows you to delete the entire table. You can use the [deleteTable\(\)](#) method of editor instance, if selection is in table. Refer to the following sample code.

```
`ts
documenteditor.editor.deleteTable();
`
```

Delete row

Document Editor allows you to delete the selected number of rows. You can use the [deleteRow\(\)](#) method of editor instance to delete the selected number of rows, if selection is in table. Refer to the following sample code.

```
`ts
documenteditor.editor.deleteRow();
`
```

Delete column

Document Editor allows you to delete the selected number of columns. You can use the [deleteColumn\(\)](#) method of editor instance to delete the selected number of columns, if selection is in table. Refer to the following sample code.

```
`ts
documenteditor.editor.deleteColumn();
`
```

Merge cells

You can merge cells vertically, horizontally, or combination of both to a single cell. To vertically merge the cells, the columns within selection should be even in left and right directions. To horizontally merge the cells, the rows within selection should be even in top and bottom direction.

Refer to the following sample code.

```
`ts
documenteditor.editor.mergeCells()
`
```

Positioning the table

Document Editor preserves the position properties of the table and displays the table based on position properties. It does not support modifying the position properties. Whereas the table will be automatically moved along with text edited if it is positioned relative to the paragraph.

How to work with tables

The following sample demonstrates how to delete the table row or columns, merge cells and how to bind the API with button.

INDEX.JS

```
var documenteditor = new ej.documenteditor.DocumentEditor({
  isReadOnly: false,
  enableSelection: true,
  enableEditorHistory: true,
  enableEditor: true,
  enableTableDialog: true,
  enableContextMenu: true,
  enableSfdtExport: true,
  height: '370px'
});
function toolbarButtonClick(arg) {
  switch (arg.item.id) {
    case 'table':
      //Insert table API to add table
      documenteditor.editor.insertTable(3, 2);
      break;
    case 'insert_above':
      //Insert the specified number of rows to the table above to the row at
      //cursor position
      documenteditor.editor.insertRow(true, 2);
      break;
    case 'insert below':
```

```

        //Insert the specified number of rows to the table below to the row at
        cursor position
        documenteditor.editor.insertRow();
        break;
    case 'insert_left':
        //Insert the specified number of columns to the table left to the
        column at cursor position
        documenteditor.editor.insertColumn(true, 2);
        break;
    case 'insert_right':
        //Insert the specified number of columns to the table right to the
        column at cursor position
        documenteditor.editor.insertColumn();
        break;
    case 'delete_table':
        //Delete the entire table
        documenteditor.editor.deleteTable();
        break;
    case 'delete_row':
        //Delete the selected number of rows
        documenteditor.editor.deleteRow();
        break;
    case 'delete_column':
        //Delete the selected number of columns
        documenteditor.editor.deleteColumn();
        break;
    case 'merge_cell':
        //Merge the selected cells into one (both vertically and horizontally)
        documenteditor.editor.mergeCells();
        break;
    case 'table_dialog':
        //Opens insert table dialog
        documenteditor.showDialog('Table');
        break;
    }
}
var toolBar = new ej.navigations.Toolbar({
    clicked: toolbarButtonClick,
    items: [
        {
            prefixIcon: 'e-de-ctnr-table e-icons',
            tooltipText: 'Insert Table',
            id: 'table',
        },
        {type: 'Separator' },
        {
            prefixIcon: 'e-de-ctnr-insertabove e-icons',
            tooltipText: 'Insert new row above',
            id: 'insert_above',
        },
        {
            prefixIcon: 'e-de-ctnr-insertbelow e-icons',
            tooltipText: 'Insert new row below',
            id: 'insert_below',
        },
        {type: 'Separator' },
        {

```

```

        prefixIcon: 'e-de-ctnr-insertleft e-icons',
        tooltipText: 'Insert new column to the left',
        id: 'insert_left',
    },
    {
        prefixIcon: 'e-de-ctnr-insertright e-icons',
        tooltipText: 'Insert new column to the right',
        id: 'insert_right',
    },
    {type: 'Separator' },
    {
        prefixIcon: 'e-icons e-de-delete-table',
        tooltipText: 'Delete Entire table',
        id: 'delete_table',
    },
    {
        prefixIcon: 'e-de-ctnr-deleterows e-icons',
        tooltipText: 'Delete the selected row',
        id: 'delete_row',
    },
    {
        prefixIcon: 'e-de-ctnr-deletecolumns e-icons',
        tooltipText: 'Delete the selected column',
        id: 'delete_column',
    },
    {type: 'Separator' },
    {
        prefixIcon: 'e-de-ctnr-mergecell e-icons',
        tooltipText: 'Merge the selected cells',
        id: 'merge_cell',
    },
    {type: 'Separator' },
    {
        text: 'Dialog',
        tooltipText: 'Open insert table dialog',
        id: 'table_dialog',
    },
    ],
    });
toolbar.appendTo('#toolbar');
documenteditor.appendTo('#DocumentEditor');
documenteditor.editor.insertTable(2, 2);

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
documenteditor/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/fabric.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="toolbar"></div>
        <div id="DocumentEditor">

            </div>
        </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Feature modules](#)
- [Insert table dialog](#)

Table of contents in ##Platform_Name## Document editor control

The table of contents in a document is same as the list of chapters at the beginning of a book. It lists each heading in the document and the page number, where that heading starts with various options to customize the appearance.

Inserting table of contents

Document Editor exposes an API to insert table of contents at cursor position programmatically. You can specify the settings for table of contents explicitly. Otherwise, the default settings will be applied.

[TableOfContentsSettings](#) contain the following properties:

- **startLevel:** Specifies the start level for constructing table of contents.
- **endLevel:** Specifies the end level for constructing table of contents.
- **includeHyperlink:** Specifies whether the link for headings is included.
- **includePageNumber:** Specified whether the page number of the headings is included.
- **rightAlign:** Specifies whether the page number is right aligned.
- **tabLeader:** Specifies the tab leader styles such as none, dot, hyphen, and underscore.
- **includeOutlineLevels:** Specifies whether the outline levels are included.

The following code illustrates how to insert table of content in Document Editor.

```
`ts
```

```
let tocSettings: TableOfContentsSettings =
```

```
{
```

```
startLevel: 1, endLevel: 3, includeHyperlink: true, includePageNumber: true, rightAlign: true
```

```
};
```

```
//Insert table of contents in Document Editor.
```

```
editor.editorModule.insertTableOfContents(tocSettings);
```

```
`
```

INDEX.TS

```
import { DocumentEditor, Editor, Selection } from '@syncfusion/ej2-
documenteditor';
//Inject require modules.
DocumentEditor.Inject(Editor, Selection);
//Initialize the Document Editor component.
let editor: DocumentEditor = new DocumentEditor({ height: '370px',
enableEditor: true, isReadOnly: false, enableSelection: true });
let documentString: string =
'{"sections":[{"blocks":[{"paragraphFormat":{"styleName":"Heading
1"},"inlines":[{"text":"Headin"}, {"name":"_GoBack", "bookmarkType":0}, {"name":
"_GoBack", "bookmarkType":1}, {"text":"g1"}]}], {"paragraphFormat":{"styleName":
"Heading
2"},"inlines":[{"text":"Heading2"}]}], {"paragraphFormat":{"styleName":"Headin
g
3"},"inlines":[{"text":"Heading3"}]}], {"paragraphFormat":{"styleName":"Headin
g
4"},"inlines":[{"text":"Heading4"}]}], {"paragraphFormat":{"styleName":"Headin
g
5"},"inlines":[{"text":"Heading5"}]}], {"paragraphFormat":{"styleName":"Headin
g
6"},"inlines":[{"text":"Heading6"}]}], {"paragraphFormat":{"styleName":"Normal
"}, "inlines":[{"text":"Normal"}]}], "headersFooters": {}, "sectionFormat": {"hea
derDistance":36.0, "footerDistance":36.0, "pageWidth":612.0, "pageHeight":792.0
, "leftMargin":72.0, "rightMargin":72.0, "topMargin":72.0, "bottomMargin":72.0, "
differentFirstPage":false, "differentOddAndEvenPages":false}}, "characterForm
at":{"fontSize":11.0, "fontFamily":"Calibri"}, "paragraphFormat":{"afterSpacin
g":8.0, "lineSpacing":1.0791666507720947, "lineSpacingType":"Multiple"}, "backg
round":{"color":"#FFFFFF"}, "styles":[{"type":"Paragraph", "name":"Normal", "
next":"Normal"}, {"type":"Paragraph", "name":"Heading
1", "basedOn":"Normal", "next":"Normal", "link":"Heading 1
```



```

Char", "characterFormat": {"fontSize": 16.0, "fontFamily": "Calibri
Light", "fontColor": "#2F5496FF"}, "paragraphFormat": {"beforeSpacing": 12.0, "aft
erSpacing": 0.0, "outlineLevel": "Level1"}}, {"type": "Paragraph", "name": "Heading
2", "basedOn": "Normal", "next": "Normal", "link": "Heading 2
Char", "characterFormat": {"fontSize": 13.0, "fontFamily": "Calibri
Light", "fontColor": "#2F5496FF"}, "paragraphFormat": {"beforeSpacing": 2.0, "afte
rSpacing": 0.0, "outlineLevel": "Level2"}}, {"type": "Paragraph", "name": "Heading
3", "basedOn": "Normal", "next": "Normal", "link": "Heading 3
Char", "characterFormat": {"fontSize": 12.0, "fontFamily": "Calibri
Light", "fontColor": "#1F3763FF"}, "paragraphFormat": {"beforeSpacing": 2.0, "afte
rSpacing": 0.0, "outlineLevel": "Level3"}}, {"type": "Paragraph", "name": "Heading
4", "basedOn": "Normal", "next": "Normal", "link": "Heading 4
Char", "characterFormat": {"italic": true, "fontFamily": "Calibri
Light", "fontColor": "#2F5496FF"}, "paragraphFormat": {"beforeSpacing": 2.0, "afte
rSpacing": 0.0, "outlineLevel": "Level4"}}, {"type": "Paragraph", "name": "Heading
5", "basedOn": "Normal", "next": "Normal", "link": "Heading 5
Char", "characterFormat": {"fontFamily": "Calibri
Light", "fontColor": "#2F5496FF"}, "paragraphFormat": {"beforeSpacing": 2.0, "afte
rSpacing": 0.0, "outlineLevel": "Level5"}}, {"type": "Paragraph", "name": "Heading
6", "basedOn": "Normal", "next": "Normal", "link": "Heading 6
Char", "characterFormat": {"fontFamily": "Calibri
Light", "fontColor": "#1F3763FF"}, "paragraphFormat": {"beforeSpacing": 2.0, "afte
rSpacing": 0.0, "outlineLevel": "Level6"}}, {"type": "Character", "name": "Default
Paragraph Font"}, {"type": "Character", "name": "Heading 1
Char", "basedOn": "Default Paragraph
Font", "characterFormat": {"fontSize": 16.0, "fontFamily": "Calibri
Light", "fontColor": "#2F5496FF"}}, {"type": "Character", "name": "Heading 2
Char", "basedOn": "Default Paragraph
Font", "characterFormat": {"fontSize": 13.0, "fontFamily": "Calibri
Light", "fontColor": "#2F5496FF"}}, {"type": "Character", "name": "Heading 3
Char", "basedOn": "Default Paragraph
Font", "characterFormat": {"fontSize": 12.0, "fontFamily": "Calibri
Light", "fontColor": "#1F3763FF"}}, {"type": "Character", "name": "Heading 4
Char", "basedOn": "Default Paragraph
Font", "characterFormat": {"italic": true, "fontFamily": "Calibri
Light", "fontColor": "#2F5496FF"}}, {"type": "Character", "name": "Heading 5
Char", "basedOn": "Default Paragraph
Font", "characterFormat": {"fontFamily": "Calibri
Light", "fontColor": "#2F5496FF"}}, {"type": "Character", "name": "Heading 6
Char", "basedOn": "Default Paragraph
Font", "characterFormat": {"fontFamily": "Calibri
Light", "fontColor": "#1F3763FF"}}}]]';
editor.appendTo('#DocumentEditor');
/*Open any existing document*/
editor.open(documentString);

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

```

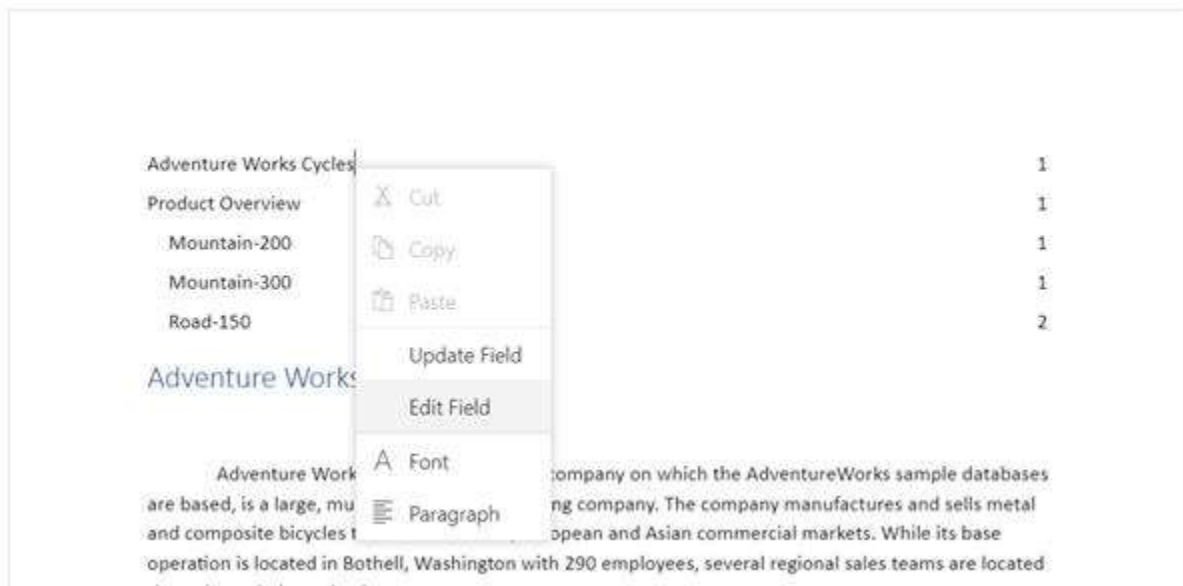
```
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
documenteditor/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div style="width:100%;height: 100%">
            <!--Element which will render as DocumentEditor -->
            <div id="DocumentEditor"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Update or edit table of contents

You can update or edit the table of contents using the built-in context menu shown up by right-clicking it. Refer to the following screenshot.



- **Update Field:** Updates the headings in table of contents with same settings by searching the entire document.
- **Edit Field:** Opens the built-in table of contents dialog and allows you to modify its settings.

You can also do it programmatically by using the exposed API. Refer to the following sample code.

```
`ts
let documentEditor: DocumentEditor = new DocumentEditor({ enableEditor: true, isReadOnly: false,
enableSelection: true });
documentEditor.appendTo('#DocumentEditor');
/Open any existing document/
editor.open("");
//Table of contents settings.
let tocSettings: TableOfContentsSettings =
{
startLevel: 1, endLevel: 3, includeHyperlink: true, includePageNumber: true, rightAlign: true
};
//Insert table of contents in Document Editor.
editor.editorModule.insertTableOfContents(tocSettings);
`
```

Same method is used for inserting, updating, and editing table of contents. This will work based on the current element at cursor position and the optional settings parameter. If table of contents is present at cursor position, the update operation will be done based on the optional settings parameter. Otherwise, the insert operation will be done.

[See Also](#)

- [Table of contents dialog](#)

Header footer in ##Platform_Name## Document editor control

Document Editor supports headers and footers in its document. Each section in the document can have the following types of headers and footers:

- First page: Used only on the first page of the section.
- Even pages: Used on all even numbered pages in the section.
- Default: Used on all pages of the section, where first or even pages are not applicable or not specified.

You can define this by setting format properties of the corresponding section using the following sample code.

```
`ts
//Defines whether different header footer is required for first page of the section
documenteditor.selection.sectionFormat.differentFirstPage = true;
//Defines whether different header footer is required for odd and even pages in the section
```

```
documenteditor.selection.sectionFormat.differentOddAndEvenPages = true;
```

```
,
```

[Go to header footer region](#)

Double click in header or footer region to move the selection into it. You can also do this by using the following code.

```
`ts
```

```
documenteditor.selection.goToHeader();
```

```
,
```

```
`ts
```

```
documenteditor.selection.goToFooter();
```

```
,
```

[Header and footer distance](#)

You can define the distance of header region content from the top of the page. Refer to the following sample code.

```
`ts
```

```
documenteditor.selection.sectionFormat.headerDistance= 36;
```

```
,
```

Same way, you can define the distance of footer region content from the bottom of the page. Refer to the following sample code.

```
`ts
```

```
documenteditor.selection.sectionFormat.footerDistace=36;
```

```
,
```

[Close header footer region](#)

Move the selection to the document body from header or footer region by double clicking or tapping the document area. You can also perform this by using the following sample code.

```
`ts
```

```
documenteditor.selection.closeHeaderFooter()
```

```
,
```

[Link to previous](#)

Link to previous is enabled by default when document has more than one section. If you're using different headers and footers such as different first page or different odd and even pages, they can't be linked together because they're all separate.

Before setting or getting the link to previous value, use the '[goToHeader](#)' or '[goToFooter](#)' API to move the current selection to the header or footer region.

You can get or set the default header footer link to previous value of a section at cursor position by using the following sample code.

```
container.documentEditor.selection.sectionFormat.oddPageHeader.linkToPrevious = false;
```

```
container.documentEditor.selection.sectionFormat.oddPageFooter.linkToPrevious = false;
`ts
```

In case the document has different header and footer types, such as different first page, odd, and even pages.

// Different first page

```
container.documentEditor.selection.sectionFormat.firstPageHeader.linkToPrevious = false;
```

```
container.documentEditor.selection.sectionFormat.firstPageFooter.linkToPrevious = false;
```

//Even page

```
container.documentEditor.selection.sectionFormat.evenPageHeader.linkToPrevious = false;
```

```
container.documentEditor.selection.sectionFormat.evenPageFooter.linkToPrevious = false;
```

`ts

Note: When there is more than one section in the document, the Link to Previous option becomes available. By default, this feature is disabled state in UI and set to return false for the first section.

See Also

- [Working with Section Formatting](#)

Text format in ##Platform_Name## Document editor control

Document Editor supports several formatting options for text like bold, italic, font color, highlight color, and more. This section describes how to modify the formatting for selected text in detail.

Bold

The bold formatting for selected text can be get or set by using the following sample code.

```
`ts
```

```
//Gets the value for bold formatting of selected text.
```

```
let bold : boolean = documenteditor.selection.characterFormat.bold;
```

```
//Sets bold formatting for selected text.
```

```
documenteditor.selection.characterFormat.bold = true;
```

`ts

You can toggle the bold formatting based on existing value at selection. Refer to the following sample code.

```
`ts
```

```
documenteditor.editor.toggleBold();
```

`ts

Italic

The Italic formatting for selected text can be get or set by using the following sample code.

```
`ts
```

```
//Gets the value for italic formatting of selected text.
let italic : boolean = documenteditor.selection.characterFormat.italic;
//Sets italic formatting for selected text.
documenteditor.selection.characterFormat.italic = true|false;
`ts
```

You can toggle the Italic formatting based on existing value at selection. Refer to the following sample code.

```
`ts
documenteditor.editor.toggleItalic();
`ts
```

Underline property

The underline style for selected text can be get or set by using the following sample code.

```
`ts
//Gets the value for underline formatting of selected text.
let underline : Underline = documenteditor.selection.characterFormat.underline;
//Sets underline formatting for selected text.
documenteditor.selection.characterFormat.underline = 'Single' | 'None';
`ts
```

You can toggle the underline style of selected text based on existing value at selection by specifying a value. Refer to the following sample code.

```
`ts
documenteditor.editor.toggleUnderline('Single');
`ts
```

Strikethrough property

The strikethrough style for selected text can be get or set by using the following sample code.

```
`ts
//Gets the value for strikethrough formatting of selected text.
let strikethrough : Strikethrough = documenteditor.selection.characterFormat.strikethrough;
//Sets strikethrough formatting for selected text.
documenteditor.selection.characterFormat.strikethrough = 'Single' | 'Normal';
`ts
```

You can toggle the strikethrough style of selected text based on existing value at selection by specifying a value. Refer to the following sample code.

```
`ts
documenteditor.editor.toggleStrikethrough();
`ts
```

,

Superscript property

The selected text can be made superscript by using the following sample code.

```
`ts
//Gets the value for baselineAlignment formatting of selected text.
let baselineAlignment : BaselineAlignment =
documenteditor.selection.characterFormat.baselineAlignment;
//Sets baselineAlignment formatting for selected text.
documenteditor.selection.characterFormat.baselineAlignment = 'Superscript';
```

,

Toggle the selected text as superscript or normal using the following sample code.

```
`ts
documenteditor.editor.toggleSuperscript();
```

,

Subscript property

The selected text can be made subscript by using the following sample code.

```
`ts
//Gets the value for baselineAlignment formatting of selected text.
let baselineAlignment : BaselineAlignment =
documenteditor.selection.characterFormat.baselineAlignment;
//Sets baselineAlignment formatting for selected text.
documenteditor.selection.characterFormat.baselineAlignment = 'Subscript';
```

,

Toggle the selected text as subscript or normal using the following sample code.

```
`ts
documenteditor.editor.toggleSubscript();
```

,

You can make a subscript or superscript text as normal using the following code.

```
`ts
documenteditor.selection.characterFormat.baselineAlignment = 'Normal';
```

,

Size

The size of selected text can be get or set using the following code.

```
`ts
//Gets the value for fontSize formatting of selected text.
```

```
let fontSize : number = documenteditor.selection.characterFormat.fontSize;
//Sets fontSize formatting for selected text.
documenteditor.selection.characterFormat.fontSize = 32;
`ts
```

Color

The color of selected text can be get or set using the following code.

```
`ts
//Gets the value for fontColor formatting of selected text.
let fontColor : string = documenteditor.selection.characterFormat.fontColor;
//Sets fontColor formatting for selected text.
documenteditor.selection.characterFormat.fontColor = 'Pink';
documenteditor.selection.characterFormat.fontColor = '#FFC0CB';
`ts
```

Font

The font style of selected text can be get or set using the following sample code.

```
`ts
//Gets the value for fontFamily formatting of selected text.
let baselineAlignment : string = documenteditor.selection.characterFormat.fontFamily;
//Sets fontFamily formatting for selected text.
documenteditor.selection.characterFormat.fontFamily = 'Arial';
`ts
```

Highlight color

The highlight color of the selected text can be get or set using the following sample code.

```
`ts
//Gets the value for highlightColor formatting of selected text.
let highlightColor : HighlightColor = documenteditor.selection.characterFormat.highlightColor;
//Sets highlightColor formatting for selected text.
documenteditor.selection.characterFormat.highlightColor = 'Pink';
documenteditor.selection.characterFormat.highlightColor = '#FFC0CB';
`ts
```

Note: 1. Character scaling and spacing present in the input Word document will be preserved in the exported Word document. N> 2. Scaling is implemented using the letterSpacing property, which may present compatibility problems. For more information, please refer to this [link](#)

Toolbar with options for text formatting

Refer to the following example.

INDEX.JS

```

var documenteditor = new ej.documenteditor.DocumentEditor({ isReadOnly:
false, enableSelection: true, enableEditorHistory: true, enableEditor: true,
enableSfdtExport: true });
function toolbarButtonClick(arg) {
    switch (arg.item.id) {
        case 'bold':
            //Toggles the bold of selected content
            documenteditor.editor.toggleBold();
            break;
        case 'italic':
            //Toggles the Italic of selected content
            documenteditor.editor.toggleItalic();
            break;
        case 'underline':
            //Toggles the underline of selected content
            documenteditor.editor.toggleUnderline('Single');
            break;
        case 'strikethrough':
            //Toggles the strikethrough of selected content
            documenteditor.editor.toggleStrikethrough();
            break;
        case 'subscript':
            //Toggles the subscript of selected content
            documenteditor.editor.toggleSubscript();
            break;
        case 'superscript':
            //Toggles the superscript of selected content
            documenteditor.editor.toggleSuperscript();
            break;
    }
}
//To change the font Style of selected content
function changeFontFamily(args) {
    documenteditor.selection.characterFormat.fontFamily = args.value;
    documenteditor.focusIn();
}
//To Change the font Size of selected content
function changeFontSize(args) {
    documenteditor.selection.characterFormat.fontSize = args.value;
    documenteditor.focusIn();
}
//To Change the font Color of selected content
function changeFontColor(args) {
    documenteditor.selection.characterFormat.fontColor =
args.currentValue.hex;
    documenteditor.focusIn();
}
documenteditor.selectionChange = () => {
    setTimeout(() => { onSelectionChange(); }, 20);
};
//Selection change to retrieve formatting
function onSelectionChange() {
    if (documenteditor.selection) {
        enableDisableFontOptions();
        // #endregion
    }
}

```

```

    }
}
function enableDisableFontOptions() {
    var characterformat = documenteditor.selection.characterFormat;
    var properties = [characterformat.bold, characterformat.italic,
characterformat.underline, characterformat.strikeThrough];
    var toggleBtnId = ["bold", "italic", "underline", "strikethrough"];
    for (var i = 0; i < properties.length; i++) {
        changeActiveState(properties[i], toggleBtnId[i]);
    }
}
function changeActiveState(property, btnId) {
    var toggleBtn = document.getElementById(btnId);
    if ((typeof (property) == 'boolean' && property == true) || (typeof
(property) == 'string' && property != 'None'))
        toggleBtn.classList.add("e-btn-toggle");
    else {
        if (toggleBtn.classList.contains("e-btn-toggle"))
            toggleBtn.classList.remove("e-btn-toggle");
    }
}
var fontStyle = ['Algerian', 'Arial', 'Calibri', 'Cambria', 'Cambria Math',
'Candara', 'Courier New', 'Georgia', 'Impact', 'Segoe Print', 'Segoe
Script', 'Segoe UI', 'Symbol', 'Times New Roman', 'Verdana', 'Windings'
];
var fontSize = ['8', '9', '10', '11', '12', '14', '16', '18',
'20', '22', '24', '26', '28', '36', '48', '72', '96'];
var toolBar = new ej.navigations.Toolbar({
    clicked: toolbarButtonClick,
    items: [
        {
            prefixIcon: 'e-de-ctnr-bold e-icons',
            tooltipText: 'Bold',
            id: 'bold',
        },
        {
            prefixIcon: 'e-de-ctnr-italic e-icons',
            tooltipText: 'Italic',
            id: 'italic',
        },
        {
            prefixIcon: 'e-de-ctnr-underline e-icons',
            tooltipText: 'Underline',
            id: 'underline',
        },
        {
            prefixIcon: 'e-de-ctnr-strikethrough e-icons',
            tooltipText: 'Strikethrough',
            id: 'strikethrough',
        },
        {
            prefixIcon: 'e-de-ctnr-subscript e-icons',
            tooltipText: 'Subscript',
            id: 'subscript',
        },
        {
            prefixIcon: 'e-de-ctnr-superscript e-icons',

```

```

        tooltipText: 'Superscript',
        id: 'superscript',
    },
    { type: 'Separator' },
    {
        type: 'Input',
        template: new ej.inputs.ColorPicker({
            value: '#000000',
            showButtons: true,
            change: changeFontColor
        }),
    },
    { type: 'Separator' },
    {
        type: 'Input',
        template: new ej.dropdowns.ComboBox({
            dataSource: fontStyle,
            width: 120,
            index: 2,
            allowCustom: true,
            change: changeFontFamily,
            showClearButton: false,
        }),
    },
    {
        type: 'Input',
        template: new ej.dropdowns.ComboBox({
            dataSource: fontSize,
            width: 80,
            allowCustom: true,
            index: 2,
            change: changeFontSize,
            showClearButton: false,
        }),
    },
    ],
});
toolBar.appendTo('#toolbar');
documenteditor.appendTo('#DocumentEditor');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
documenteditor/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/fabric.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="toolbar">
        </div>
        <div style="width:100%;height: 100%">
            <!--Element which will render as DocumentEditor -->
            <div id="DocumentEditor"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Feature modules](#)
- [Font dialog](#)
- [Keyboard shortcuts](#)

Paragraph format in ##Platform_Name## Document editor control

Document Editor supports various paragraph formatting options such as text alignment, indentation, paragraph spacing, and more.

Indentation

You can modify the left or right indentation of selected paragraphs using the following sample code.

```
`ts
```

```
documenteditor.selection.paragraphFormat.leftIndent = 24;
```

```
documenteditor.selection.paragraphFormat.rightIndent = 24;
```

```
,
```

Special indentation

You can define special indent for first line of the paragraph using the following sample code.

```
`ts
```

```
documenteditor.selection.paragraphFormat.firstLineIndent = 24;
```

```
,
```

Increase indent

You can increase the left indent of selected paragraphs by a factor of 36 points using the following sample code.

```
`ts
```

```
documenteditor.editor.increaseIndent()
```

```
,
```

Decrease indent

You can decrease the left indent of selected paragraphs by a factor of 36 points using the following sample code.

```
`ts
```

```
documenteditor.editor.decreaseIndent()
```

```
,
```

Text alignment

You can get or set the text alignment of selected paragraphs using the following sample code.

```
`ts
```

```
documenteditor.selection.paragraphFormat.textAlignment = 'Center' | 'Left' | 'Right' | 'Justify';
```

```
,
```

Note: Starting from v19.4.0.x, the text justification of Document editor component matches alignment of Microsoft Word 2013 and newer versions based on the compatibility mode present in the document. The DOCX document created using Microsoft Word 2013 and newer versions will have the compatibility mode **Word2013** and follows a special behavior in justifying the text. You can retain the text justification behavior like old versions by modifying the compatibility mode as **Word2010**.

```
`ts
```

```
documenteditor.documentSettings.compatibilityMode = 'Word2010';
```

```
,
```

Note: The Document editor component assumes the compatibility mode as **Word2013** by default, if it is not defined for a document.

Microsoft Word 2010 and older versions**Justified**

Giant pandas can digest bamboo is attributed to tiny microbes that live within their digestive system. As they can only digest about 20% of what they eat, the average giant panda consumes around 14 kilograms (30 pounds) of bamboo a day. In comparison, humans eat about 2 kilograms (5 pounds) of food a day

Microsoft Word 2013 and newer versions**Justified**

Giant pandas can digest bamboo is attributed to tiny microbes that live within their digestive system. As they can only digest about 20% of what they eat, the average giant panda consumes around 14 kilograms (30 pounds) of bamboo a day. In comparison, humans eat about 2 kilograms (5 pounds) of food a day

You can toggle the text alignment of selected paragraphs by specifying a value using the following sample code.

```
`ts
```

```
documenteditor.editor.toggleTextAlignment('Center' | 'Left' | 'Right' | 'Justify');
```

```
,
```

Line spacing and its type

You can define the line spacing and its type for selected paragraphs using the following sample code.

```
`ts
```

```
documenteditor.selection.paragraphFormat.lineSpacingType = 'AtLeast';
```

```
documenteditor.selection.paragraphFormat.lineSpacing = 6;
```

```
,
```

Paragraph spacing

You can define the spacing before or after the paragraph by using the following sample code.

```
`ts
```

```
documenteditor.selection.paragraphFormat.beforeSpacing = 24;
```

```
documenteditor.selection.paragraphFormat.afterSpacing = 24;
```

```
,
```

You can also set automatic spacing before and after the paragraph by using the following sample code.

```
`ts
```

```
documenteditor.selection.paragraphFormat.spaceBeforeAuto = true;
```

```
documenteditor.selection.paragraphFormat.spaceAfterAuto = true;
```

```
,
```

Note: If auto spacing property is enabled, then value defined in the `beforeSpacing` and `afterSpacing` property will not be considered.

Pagination properties

You can enable or disable the following pagination properties for the paragraphs in a Word document.

- Widow/Orphan control - whether the first and last lines of the paragraph are to remain on the same page as the rest of the paragraph when paginating the document.
- Keep with next - whether the specified paragraph remains on the same page as the paragraph that follows it while paginating the document.
- Keep lines together - whether all lines in the specified paragraphs remain on the same page while paginating the document.

The following example code illustrates how to enable or disable these pagination properties for the selected paragraphs.

```
`ts
documenteditor.selection.paragraphFormat.widowControl = false;
documenteditor.selection.paragraphFormat.keepWithNext = true;
documenteditor.selection.paragraphFormat.keepLinesTogether = true;
`
```

Paragraph Border

You can apply borders to the paragraphs in a Word document. Using borders, decorate the paragraphs to set them apart from other paragraphs in the document.

The following example code illustrates how to apply box border for the selected paragraphs.

```
`ts
// left
documenteditor.selection.paragraphFormat.borders.left.lineStyle = 'Single';
documenteditor.selection.paragraphFormat.borders.left.lineWidth = 3;
documenteditor.selection.paragraphFormat.borders.left.color = "#000000";
//right
documenteditor.selection.paragraphFormat.borders.right.lineStyle = 'Single';
documenteditor.selection.paragraphFormat.borders.right.lineWidth = 3;
documenteditor.selection.paragraphFormat.borders.right.color = "#000000";
//top
documenteditor.selection.paragraphFormat.borders.top.lineStyle = 'Single';
documenteditor.selection.paragraphFormat.borders.top.lineWidth = 3;
documenteditor.selection.paragraphFormat.borders.top.color = "#000000";
//bottom
```

```
documenteditor.selection.paragraphFormat.borders.bottom.lineStyle = 'Single';
documenteditor.selection.paragraphFormat.borders.bottom.lineWidth = 3;
documenteditor.selection.paragraphFormat.borders.bottom.color = "#000000";
`ts
```

Note: At present, the Document editor component displays all the border styles as single line. But you can apply any border style and get the proper display in Microsoft Word app when opening the exported Word document.

Show or Hide Paragraph marks

You can show or hide the hidden formatting symbols like spaces, tab, paragraph marks, and breaks in Document editor component. These marks help identify the start and end of a paragraph and all the hidden formatting symbols in a Word document.

The following example code illustrates how to show or hide paragraph marks.

```
`ts
documenteditor.documentEditorSettings.showHiddenMarks = true;
`ts
```

Toolbar with paragraph formatting options

The following sample demonstrates the paragraph formatting options using a toolbar.

INDEX.JS

```
var documenteditor = new ej.documenteditor.DocumentEditor({
  isReadOnly: false, enableSelection: true, enableEditorHistory: true,
  enableEditor: true, enableContextMenu: true, enableSfdtExport: true,height:
  '370px'
});
function toolbarButtonClick(arg) {
  switch (arg.item.id) {
    case 'AlignLeft':
      //Toggle the Left alignment for selected or current paragraph
      documenteditor.editor.toggleTextAlignment('Left');
      break;
    case 'AlignRight':
      //Toggle the Right alignment for selected or current paragraph
      documenteditor.editor.toggleTextAlignment('Right');
      break;
    case 'AlignCenter':
      //Toggle the Center alignment for selected or current paragraph
      documenteditor.editor.toggleTextAlignment('Center');
      break;
    case 'Justify':
      //Toggle the Justify alignment for selected or current paragraph
      documenteditor.editor.toggleTextAlignment('Justify');
      break;
    case 'IncreaseIndent':
      //Increase the left indent of selected or current paragraph
      documenteditor.editor.increaseIndent();
      break;
    case 'DecreaseIndent':
      //Decrease the left indent of selected or current paragraph
```



```

        documenteditor.editor.decreaseIndent();
        break;
    case 'ClearFormat':
        documenteditor.editor.clearFormatting();
        break;
    }
}
//Change the line spacing of selected or current paragraph
function lineSpacingAction(args) {
    var text = args.item.text;
    switch (text) {
        case 'Single':
            documenteditor.selection.paragraphFormat.lineSpacing = 1;
            break;
        case '1.15':
            documenteditor.selection.paragraphFormat.lineSpacing = 1.15;
            break;
        case '1.5':
            documenteditor.selection.paragraphFormat.lineSpacing = 1.5;
            break;
        case 'Double':
            documenteditor.selection.paragraphFormat.lineSpacing = 2;
            break;
    }
    setTimeout(function () {
        documenteditor.focusIn();
    }, 30);
}
documenteditor.selectionChange = () => {
    setTimeout(() => {
        onSelectionChange();
    }, 20);
};
// Selection change to retrieve formatting
function onSelectionChange() {
    if (documenteditor.selection) {
        var paragraphFormat = documenteditor.selection.paragraphFormat;
        var toggleBtnId = ['AlignLeft', 'AlignCenter', 'AlignRight', 'Justify'];
        for (var i = 0; i < toggleBtnId.length; i++) {
            var toggleBtn = document.getElementById(
                toggleBtnId[i]
            );
            toggleBtn.classList.remove('e-btn-toggle');
        }
        if (paragraphFormat.textAlignment === 'Left') {
            document.getElementById('AlignLeft').classList.add('e-btn-toggle');
        } else if (paragraphFormat.textAlignment === 'Right') {
            document.getElementById('AlignRight').classList.add('e-btn-toggle');
        } else if (paragraphFormat.textAlignment === 'Center') {
            document
                .getElementById('AlignCenter')
                .classList.add('e-btn-toggle');
        } else {
            document.getElementById('Justify').classList.add('e-btn-toggle');
        }
    }
    // #endregion
}

```

```

}
//Toolbar configuration to add paragraph formatting options
var toolBar = new ej.navigations.Toolbar({
  clicked: toolbarButtonClick,
  items: [
    {
      prefixIcon: 'e-de-ctnr-alignleft e-icons',
      tooltipText: 'Align Left',
      id: 'AlignLeft',
    },
    {
      prefixIcon: 'e-de-ctnr-aligncenter e-icons',
      tooltipText: 'Align Center',
      id: 'AlignCenter',
    },
    {
      prefixIcon: 'e-de-ctnr-alignright e-icons',
      tooltipText: 'Align Right',
      id: 'AlignRight',
    },
    {
      prefixIcon: 'e-de-ctnr-justify e-icons',
      tooltipText: 'Justify',
      id: 'Justify',
    },
    {
      prefixIcon: 'e-de-ctnr-increaseindent e-icons',
      tooltipText: 'Increase Indent',
      id: 'IncreaseIndent',
    },
    {
      prefixIcon: 'e-de-ctnr-decreaseindent e-icons',
      tooltipText: 'Decrease Indent',
      id: 'DecreaseIndent',
    },
    { type: 'Separator' },
    {
      id: 'lineSpacing',
    },
    {
      prefixIcon: 'e-de-ctnr-clearall e-icons',
      tooltipText: 'ClearFormatting',
      id: 'ClearFormat',
    }
  ],
});
toolBar.appendTo('#toolbar');
var items = [
  {
    text: 'Single',
  },
  {
    text: '1.15',
  },
  {
    text: '1.5',
  },
  {

```

```

        text: 'Double',
    },
];
var dropdown = new ej.splitbuttons.DropDownButton({
    items: items,
    iconCss: 'e-de-ctnr-linespacing e-icons',
    select: lineSpacingAction,
});
dropdown.appendTo('#lineSpacing');
documenteditor.appendTo('#DocumentEditor');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
documenteditor/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/fabric.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="toolbar">
        </div>
        <div style="width:100%;height: 100%">
            <!--Element which will render as DocumentEditor -->
            <div id="DocumentEditor"></div>
        </div>
    </div>
</body>
</html>

```

```
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

See Also

- [Feature modules](#)
- [Paragraph dialog](#)
- [Keyboard shortcuts](#)

List format in ##Platform_Name## Document editor control

Document Editor supports both the single-level and multilevel lists. Lists are used to organize data as step-by-step instructions in documents for easy understanding of key points. You can apply list to the paragraph either using supported APIs.

Create bullet list

Bullets are usually used for unordered lists. To apply bulleted list for selected paragraphs, use the following method of ‘Editor’ instance.

applyBullet(bullet, fontFamily);

Parameter	Type	Description
bullet	string	Bullet character.
fontFamily	string	Bullet font family.

Refer to the following sample code.

```
`ts
```

```
documenteditor.editor.applyBullet('\uf0b7', 'Symbol');
```

```
,
```

Create numbered list

Numbered lists are usually used for ordered lists. To apply numbered list for selected paragraphs, use the following method of ‘Editor’ instance.

applyNumbering(numberFormat,listLevelPattern)

Parameter	Type	Description
numberFormat	string	“%n” representations in ‘numberFormat’ parameter will be replaced by respective list level’s value. “%1” will be displayed as “1”
listLevelPattern(optional)	string	Default value is 'Arabic'.

Refer to the following example.

```
`ts
documenteditor.editor.applyNumbering('%1', 'UpRoman');
`
```

Clear list

You can also clear the list formatting applied for selected paragraphs. Refer to the following sample code.

```
`ts
documenteditor.editor.clearList();
`
```

Working with lists

The following sample demonstrates how to create bullet and numbering lists in Document Editor.

INDEX.JS

```
ej.base.enableRipple(true);
ej.documenteditor.DocumentEditor.Inject(ej.documenteditor.Editor,
ej.documenteditor.Selection,ej.documenteditor.EditorHistory);
var documenteditor = new ej.documenteditor.DocumentEditor({ enableEditor:
true, isReadOnly: false, enableSelection:
true,enableEditorHistory:true,height: '370px'});
function toolbarAction (args){
  switch (args.item.id) {
    case 'Bullets':
      //To create bullet list
      documenteditor.editor.applyBullet('\uf0b7', 'Symbol');
      break;
    case 'Numbering':
      //To create numbering list
      documenteditor.editor.applyNumbering('%1', 'UpRoman');
      break;
    case 'clearlist':
      //To clear list
      documenteditor.editor.clearList();
      break;
  }
};
var toolbar = new ej.navigations.Toolbar({
  clicked: toolbarAction,
  items: [
    {
      prefixIcon: 'e-de-ctnr-bullets e-icons',
      tooltipText: 'Bullets',
      id: 'Bullets',
    },
    {
      prefixIcon: 'e-de-ctnr-numbering e-icons',
      tooltipText: 'Numbering',
      id: 'Numbering',
    },
    {
      text: 'Clear',
    }
  ]
});
```

```

        id: 'clearlist',
        tooltipText: 'Clear List',
    },
],
});
toolbar.appendTo('#toolbar');
documenteditor.appendTo('#DocumentEditor');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
documenteditor/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/fabric.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

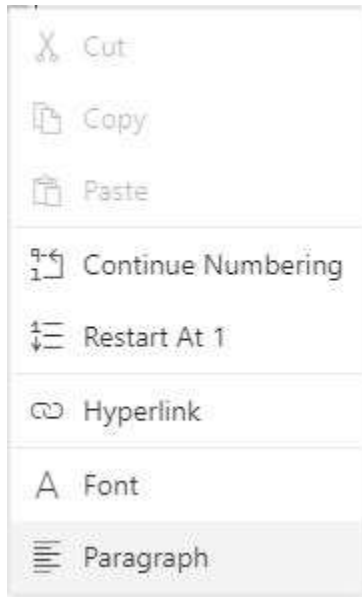
    <div id="container">
        <div id="toolbar">
        </div>
        <div style="width:100%;height: 100%">
            <!--Element which will render as DocumentEditor -->
            <div id="DocumentEditor"></div>
        </div>
    </div>
</body>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";

```

```
}  
    </script>  
<script src="index.js" type="text/javascript"></script>  
</body></html>
```

Editing numbered list

Document Editor restarts the numbering or continue numbering for a numbered list. These options are found in the built-in context menu, if the list value is selected. Refer to the following screenshot.



See Also

- [List dialog](#)

Table format in ##Platform_Name## Document editor control

Document Editor customizes the formatting of table, or table cells such as table width, cell margins, cell spacing, background color, and table alignment. This section describes how to customize these formatting for selected cells, rows, or table in detail.

Cell margins

You can customize the cell margins by using the following sample code.

```
`ts  
//To change the left margin  
documenteditor.selection.cellFormat.leftMargin = 5.4;  
//To change the right margin  
documenteditor.selection.cellFormat.rightMargin = 5.4;  
//To change the top margin  
documenteditor.selection.cellFormat.topMargin = 5.4;  
//To change the bottom margin
```

```
documenteditor.selection.cellFormat.bottomMargin = 5.4;
```

```
,
```

You can also define the default cell margins for a table. If the specific cell margin value is not defined explicitly in the cell formatting, the corresponding value will be retrieved from default cells margin of the table. Refer to the following sample code.

```
`ts
```

```
//To change the left margin
```

```
documenteditor.selection.tableFormat.leftMargin = 5.4;
```

```
//To change the right margin
```

```
documenteditor.selection.tableFormat.rightMargin = 5.4;
```

```
//To change the top margin
```

```
documenteditor.selection.tableFormat.topMargin = 5.4;
```

```
//To change the bottom margin
```

```
documenteditor.selection.tableFormat.bottomMargin = 5.4;
```

```
,
```

Background color

You can explicitly set the background color of selected cells using the following sample code.

```
`ts
```

```
documenteditor.selection.cellFormat.background = '#E0E0E0';
```

```
,
```

Refer to the following sample code to customize the background color of the table.

```
`ts
```

```
documenteditor.selection.tableFormat.background = '#E0E0E0';
```

```
,
```

Cell spacing

Refer to the following sample code to customize the spacing between each cell in a table.

```
`ts
```

```
documenteditor.selection.tableFormat.cellSpacing = 2;
```

```
,
```

Cell vertical alignment

The content is aligned within a table cell to 'Top', 'Center', or 'Bottom'. You can customize this property of selected cells. Refer to the following sample code.

```
`ts
```

```
documenteditor.selection.cellFormat.verticalAlignment = 'Bottom';
```

```
,
```


Table alignment

The tables are aligned in Document Editor to 'Left', 'Right', or 'Center'. Refer to the following sample code.

```
`ts
documenteditor.selection.tableFormat.tableAlignment = 'Center';
`
```

Cell width

Set the desired width of table cells that will be considered when the table is layouted. Refer to the following sample code.

```
`ts
import { DocumentEditor, Editor, Selection, SfdtExport } from '@syncfusion/ej2-documenteditor';
//Inject the required module
DocumentEditor.Inject(Editor, Selection, SfdtExport);
let documenteditor: DocumentEditor = new DocumentEditor({
isReadOnly: false,
enableSelection: true,
enableEditor: true,
enableSfdtExport: true
});
documenteditor.appendTo('#DocumentEditor');
documenteditor.editor.insertTable(2, 2);
//To change the width of a cell
documenteditor.selection.cellFormat.preferredWidthType = 'Point';
documenteditor.selection.cellFormat.preferredWidth = 100;
`
```

Table width

You can set the desired width of a table in 'Point 'or 'Percent' type. Refer to the following sample code.

```
`ts
import { DocumentEditor, Editor, Selection, SfdtExport } from '@syncfusion/ej2-documenteditor';
//Inject the required module
DocumentEditor.Inject(Editor, Selection, SfdtExport);
let documenteditor: DocumentEditor = new DocumentEditor({
isReadOnly: false,
enableSelection: true,
enableEditor: true,
```

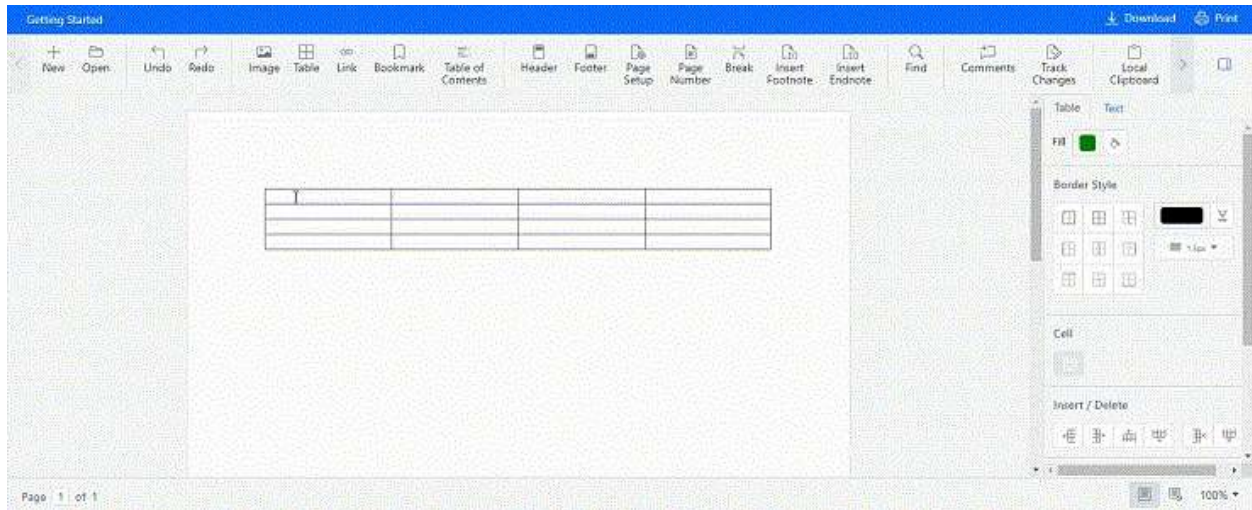
```
enableSfdtExport: true
});
documenteditor.appendTo('#DocumentEditor');
documenteditor.editor.insertTable(2, 2);
//To change the width of a table
documenteditor.selection.tableFormat.preferredWidthType = 'Point';
documenteditor.selection.tableFormat.preferredWidth = 300;
`
```

Apply borders

Document Editor exposes API to customize the borders for table cells by specifying the settings. Refer to the following sample code.

```
`ts
import { DocumentEditor, Editor, Selection, SfdtExport, BorderSettings } from '@syncfusion/ej2-
documenteditor';
//Inject the required module
DocumentEditor.Inject(Editor, Selection, SfdtExport);
let documenteditor: DocumentEditor = new DocumentEditor({
isReadOnly: false,
enableSelection: true,
enableEditor: true,
enableSfdtExport: true
});
documenteditor.appendTo('#DocumentEditor');
documenteditor.editor.insertTable(2, 2);
//To apply border
let borderSettings: BorderSettings = {
type: 'AllBorders',
lineWidth: 12
};
documenteditor.editor.applyBorders(borderSettings);
`
```

Please check below gif which illustrates how to apply border for selected cells through properties pane options - border color, line size and no border:



Working with row formatting

Document Editor allows various row formatting such as height and repeat header.

Row height

You can customize the height of a table row as 'Auto', 'AtLeast', or 'Exactly'. Refer to the following sample code.

```
`ts
```

```
import { DocumentEditor, Editor, Selection, SfdtExport } from '@syncfusion/ej2-documenteditor';
```

```
//Inject the required module
```

```
DocumentEditor.Inject(Editor, Selection, SfdtExport);
```

```
let documenteditor: DocumentEditor = new DocumentEditor({
```

```
  isReadOnly: false,
```

```
  enableSelection: true,
```

```
  enableEditor: true,
```

```
  enableSfdtExport: true
```

```
});
```

```
documenteditor.appendTo('#DocumentEditor');
```

```
documenteditor.editor.insertTable(2, 2);
```

```
//To change row height of first row
```

```
documenteditor.selection.rowFormat.heightType = 'Exactly';
```

```
documenteditor.selection.rowFormat.height = 20;
```

```
,
```

Header row

The header row describes the content of a table. A table can optionally have a header row. Only the first row of a table can be the header row. If the cursor position is at first row of the table, then you can define whether it as header row or not, using the following sample code.

```
`ts
```

```
documenteditor.selection.rowFormat.isHeader = true;
```

```
,
```

Allow row break across pages

This property is valid if a table row does not fit in the current page during table layout. It defines whether a table row can be allowed to break. If the value is false, the entire row will be moved to the start of next page. You can modify this property for selected rows using the following sample code.

```
`ts
```

```
documenteditor.selection.rowFormat.allowRowBreakAcrossPages = false;
```

```
,
```

Title

Document Editor expose API to get or set the table title of the selected table. Refer to the following sample code to set title.

```
`ts
```

```
documenteditor.selection.tableFormat.title = 'Shipping Details';
```

```
,
```

Description

Document Editor expose API to get or set the table description of the selected image. Refer to the following sample code to set description.

```
`ts
```

```
documenteditor.selection.tableFormat.description = 'Freight cost and shipping details';
```

```
,
```

See Also

- [Table properties dialog](#)

Section format in ##Platform_Name## Document editor control

Document Editor supports various section formatting such as page size, page margins, and more.

Page size

You can get or set the size of a section at cursor position by using the following sample code.

```
`ts
```

```
documenteditor.selection.sectionFormat.pageWidth = 500;
```

```
documenteditor.selection.sectionFormat.pageHeight = 600;
```

```
,
```

You can change the orientation of the page by swapping the values of page width and height respectively.

Page margins

Left and right page margin defines the gap between the document content from left and right side of the page respectively. Top and bottom page margins defines the gap between the document content from header and footer of the page respectively.

Refer to the following sample code.

```
`ts
documenteditor.selection.sectionFormat.leftMargin = 10;
documenteditor.selection.sectionFormat.rightMargin = 10;
documenteditor.selection.sectionFormat.bottomMargin = 10;
documenteditor.selection.sectionFormat.topMargin = 10;
`
```

Header distance

You can define the distance of header content from the top of the page by using the following sample code.

```
`ts
documenteditor.selection.sectionFormat.headerDistance = 72;
`
```

Footer distance

You can define the distance of footer content from the bottom of the page by using the following sample code.

```
`ts
documenteditor.selection.sectionFormat.footerDistance = 72;
`
```

Columns

You can define the number of columns, column width, and space between columns for the pages in a section.

The following code example illustrates how to define the two columns layout for the pages in a section.

```
`ts
let column: SelectionColumnFormat = new
SelectionColumnFormat(container.documentEditor.selection);
column.width = 216;
column.space = 36;
container.documentEditor.selection.sectionFormat.columns = [column, column];
container.documentEditor.selection.sectionFormat.lineBetweenColumns = true;
`
```

Breaks

You can insert column break. The following code example illustrates how to insert a column break.

```
`ts
container.documentEditor.editor.insertColumnBreak();
`
```

You can insert next page section break to start the new section on the next page.

The following code example illustrates how to insert a next page section break.

```
`ts
container.documentEditor.editor.insertSectionBreak(SectionBreakType.NewPage);
`
```

You can insert continuous section break to start the new section on the same page.

The following code example illustrates how to insert a continuous section break.

```
`ts
container.documentEditor.editor.insertSectionBreak(SectionBreakType.Continuous);
`
```

See Also

- [Page setup dialog](#)
- [Column dialog](#)

Comments in ##Platform_Name## Document editor control

Document Editor allows you to add comments to documents. You can add, navigate and remove comments in code and from the UI.

Add a new comment

Comments can be inserted to the selected text.

```
`ts
//Add new commnt in the document.
documentEditor.editor.insertComment("Test comment");
`
```

Comment navigation

Next and previous comments can be navigated using the below code snippet.

```
`ts
//Navigate to next comment
documentEditor.selection.navigateNextComment();

//Navigate to previous comment
documentEditor.selection.navigatePreviousComment();
`
```

,

Delete comment

Current comment can be deleted using the below code snippet.

```
`ts
//Delete the selected comment.
documentEditor.editor.deleteComment();
,
```

Delete all comment

All the comments in the document can be deleted using the below code snippet.

```
`ts
//Delete all the comments present in the current document.
documentEditor.editor.deleteAllComments();
,
```

Protect the document in comments only mode

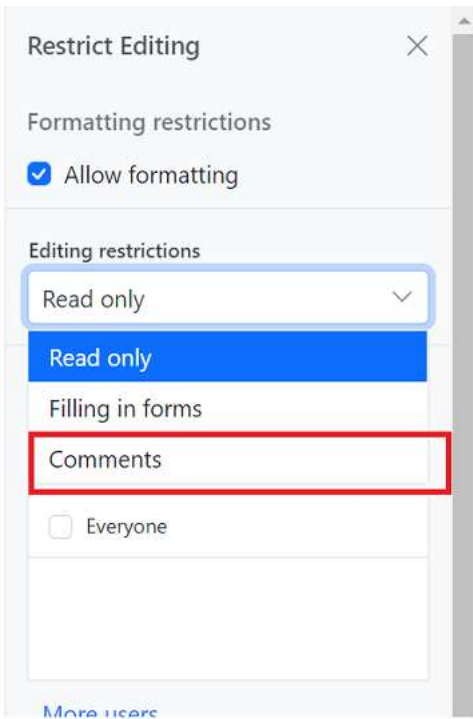
Document Editor provides support for protecting the document with **CommentsOnly** protection. In this protection, user allowed to add or edit comments alone in the document.

Document editor provides an option to protect and unprotect document using [enforceProtection](#) and [stopProtection](#) API.

The following example code illustrates how to enforce and stop protection in Document editor container.

```
`ts
let container: DocumentEditorContainer = new DocumentEditorContainer({
enableToolbar: true,
height: '590px',
});
DocumentEditorContainer.Inject(Toolbar);
container.serviceUrl =
'http://localhost:5000/api/documenteditor/';
container.appendTo('#container');
//enforce protection
container.documentEditor.editor.enforceProtection('123', 'CommentsOnly');
//stop the document protection
container.documentEditor.editor.stopProtection('123');
,
```

Comment only protection can be enabled in UI by using [Restrict Editing pane](#)



Note: In enforce Protection method, first parameter denotes password and second parameter denotes protection type. Possible values of protection type are `NoProtection` | `ReadOnly` | `FormFieldsOnly` | `CommentsOnly`. In stop protection method, parameter denotes the password.

Fields in ##Platform_Name## Document editor control

Document Editor has preservation support for all types of fields in an existing word document without any data loss.

Adding Fields

You can add a field to the document by using [insertField](#) method in [Editor](#) module.

The following example code illustrates how to insert merge field programmatically by providing the field code and field result.

```
`ts
let fieldCode: string = 'MERGEFIELD First Name \* MERGEFORMAT ';
let fieldResult: string = '«First Name»';
documenteditor.editor.insertField(fieldCode, fieldResult);
`
```

Note: Document editor does not validate/process the field code/field result. it simply inserts the field with specified field information.

Update fields

Document Editor provides support for updating bookmark cross reference field. The following example code illustrates how to update bookmark cross reference field.

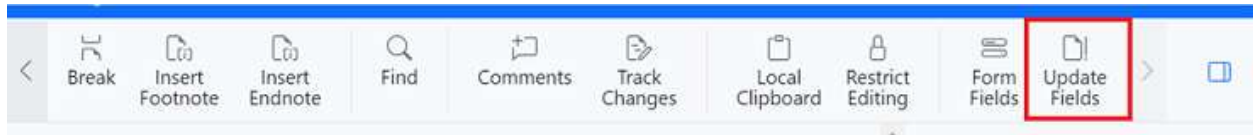
```
`ts
```


//Update all the bookmark cross reference field in the document.

```
documentEditor.updateFields();
```

,

Bookmark cross reference fields can be updated through UI by using update fields option in **Toolbar**.



The following type of fields are automatically updated in Document Editor.

- Numpages
- Section
- Page

Get field info

You can get field code and field result of the current selected field by using [getFieldInfo](#) method in the [Selection](#) module.

```
`ts
```

```
//Gets the field information of the selected field.
```

```
let fieldInfo: FieldInfo = documentEditor.selection.getFieldInfo();
```

,

Note: For nested fields, this method returns combined field code and result.

Set field info

You can modify the field code and field result of the current selected field by using [setFieldInfo](#) method in the [Editor](#) module.

```
`ts
```

```
//Gets the field information for the selected field.
```

```
let fieldInfo: FieldInfo = documentEditor.selection.getFieldInfo();
```

```
//Modify field code
```

```
fieldInfo.code = 'MERGEFIELD First Name \* MERGEFORMAT';
```

```
//Modify field result
```

```
fieldInfo.result = '«First Name»';
```

```
//Modify field code and result of the current selected field.
```

```
documentEditor.editor.setFieldInfo(fieldInfo);
```

,

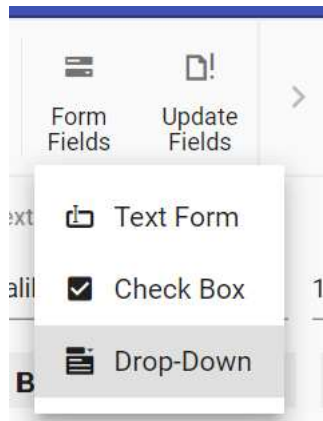
Note: For nested field, entire field gets replaced completely with the specified field information.

See Also

- [Mail merge using DocIO](#)
- [Mail merge demo](#)
- You can refer to the [Microsoft support article to know more about the list of fields supported in Microsoft Word and its field codes](#)

Form fields in ##Platform_Name## Document editor control

DocumentEditorContainer component provide support for inserting Text, CheckBox, DropDown form fields through in-built toolbar.



Insert form field

Form fields can be inserted using [insertFormField](#) method in editor module.

```
`ts
//Insert Text form field
documentEditor.editor.insertFormField('Text');
//Insert Checkbox form field
documentEditor.editor.insertFormField('CheckBox');
//Insert Drop down form field
documentEditor.editor.insertFormField('Dropdown');
`
```

Get form field names

All the form fields names form current document can be retrieved using [getFormFieldNames\(\)](#).

```
`ts
let formFieldsNames: string[] = documentEditor.getFormFieldNames();
`
```

Get form field properties

Form field properties can be retrieved using [getFormFieldInfo\(\)](#).

```
`ts
```

```
//Get Text form field by using bookmark name.
```

```
let textfieldInfo: TextFormFieldInfo = documentEditor.getFormFieldInfo('Text1') as TextFormFieldInfo;
```

```
//Checkbox form field by using bookmark name.
```

```
let checkboxfieldInfo: CheckBoxFormFieldInfo = documentEditor.getFormFieldInfo('Check1') as
CheckBoxFormFieldInfo;
```

```
//Dropdown form field by using bookmark name.
```

```
let dropdownfieldInfo: DropDownFormFieldInfo = documentEditor.getFormFieldInfo('Drop1') as
DropDownFormFieldInfo;
```

```
,
```

Set form field properties

Form field properties can be modified using [setFormFieldInfo](#).

```
`ts
```

```
// Set text form field properties
```

```
let textfieldInfo: TextFormFieldInfo = documentEditor.getFormFieldInfo('Text1') as TextFormFieldInfo;
```

```
textfieldInfo.defaultValue = "Hello";
```

```
textfieldInfo.format = "Uppercase";
```

```
textfieldInfo.type = "Text";
```

```
textfieldInfo.name = "Text2";
```

```
documentEditor.setFormFieldInfo('Text1', textfieldInfo);
```

```
// Set checkbox form field properties
```

```
let checkboxfieldInfo: CheckBoxFormFieldInfo = documentEditor.getFormFieldInfo('Check1') as
CheckBoxFormFieldInfo;
```

```
checkboxboxfieldInfo.defaultValue = true;
```

```
checkboxboxfieldInfo.name = "Check2";
```

```
documentEditor.setFormFieldInfo('Check1', checkboxfieldInfo);
```

```
// Set checkbox form field properties
```

```
let dropdownfieldInfo: DropDownFormFieldInfo = documentEditor.getFormFieldInfo('Drop1') as
DropDownFormFieldInfo;
```

```
dropdownfieldInfo.dropDownItems = ['One', 'Two', 'Three']
```

```
dropdownfieldInfo.name = "Drop2";
```

```
documentEditor.setFormFieldInfo('Drop1', dropdownfieldInfo);
```

```
,
```

Note:If a form field already exists in the document with the new name specified, the old form field name property will be cleared and it will not be accessible. Ensure the new name is unique.

Export form field data

Data of the all the Form fields in the document can be exported using [exportFormData](#).

```
`ts
```

```
let formFieldDate: FormFieldData[] = documentEditor.exportFormData();
```

```
,
```

Import form field data

Form fields can be prefilled with data using [importFormData](#).

```
`ts
```

```
let textformField: FormFieldData = { fieldName: 'Text1', value: 'Hello World' };
```

```
let checkformField: FormFieldData = { fieldName: 'Check1', value: true };
```

```
let dropdownformField: FormFieldData = { fieldName: 'Drop1', value: 1 };
```

```
//Import form field data
```

```
this.container.documentEditor.importFormData([textformField, checkformField, dropdownformField]);
```

```
,
```

Reset form fields

Reset all the form fields in current document to default value using [resetFormFields](#).

```
`ts
```

```
documentEditor.resetFormFields();
```

```
,
```

Protect the document in form filling mode

Document Editor provides support for protecting the document with **FormFieldsOnly** protection. In this protection, user can only fill form fields in the document.

Document editor provides an option to protect and unprotect document using [enforceProtection](#) and [stopProtection](#) API.

The following example code illustrates how to enforce and stop protection in Document editor container.

```
`ts
```

```
let container: DocumentEditorContainer = new DocumentEditorContainer({
```

```
enableToolbar: true,
```

```
height: '590px',
```

```
});
```

```
DocumentEditorContainer.Inject(Toolbar);
```

```
container.serviceUrl =
```

```
'https://services.syncfusion.com/js/production/api/documenteditor/';
```

```
container.appendTo('#container');
```

```
//enforce protection
container.documentEditor.editor.enforceProtection('123', 'FormFieldsOnly');
//stop the document protection
container.documentEditor.editor.stopProtection('123');
```

Note: In enforce Protection method, first parameter denotes password and second parameter denotes protection type. Possible values of protection type are NoProtection | ReadOnly | FormFieldsOnly | CommentsOnly. In stop protection method, parameter denotes the password.

Clipboard in ##Platform_Name## Document editor control

Document Editor takes advantage of system clipboard and allows you to copy or move a portion of the document into it in HTML format, so that it can be pasted in any application that supports clipboard.

Copy

Copy a portion of document to system clipboard using built-in context menu of Document Editor. You can also do it programmatically using the following sample code.

```
`ts
documentEditor.selection.copy();
```

Cut

Cut a portion of document to system clipboard using built-in context menu of Document Editor. You can also do it programmatically using the following sample code.

```
`ts
documentEditor.editor.cut();
```

Paste

Due to limitations, you can paste contents from system clipboard in Document Editor only using the 'CTRL + V' keyboard shortcut.

Note: Due to browser limitation of getting content from system clipboard, paste using API and context menu option doesn't work.

Local paste (copy/paste within control)

Document Editor expose API to enable local paste within the control. On enabling this, the following is performed:

- Selected contents will be stored to an internal clipboard in addition to system clipboard.
- Clipboard paste will be overridden, and internally stored data (SFDT data) that has formatted text will be pasted using paste() API in Document editor.

Refer to the following sample code.

```
`ts
```

```
//Initialize the Document Editor.
```

```
let editor: DocumentEditor = new DocumentEditor({ enableEditor: true, isReadOnly: false,
enableSelection: true });
```

```
//Enable the local paste.
```

```
editor.enableLocalPaste = true;
```

```
,
```

By default, **enableLocalPaste** is false.

When local paste is enabled for a Document Editor instance, you can paste contents programmatically if the internal clipboard has stored data during last copy operation. Refer to the following sample code.

```
`ts
```

```
documentEditor.editor.paste();
```

```
,
```

Paste options in context menu

In Document editor, paste options in context menu will be in disabled state if you were try to copy/paste content from outside of Document editor. It gets enabled when **enableLocalPaste** is true and trying to copy/paste content inside Document editor.

Note: Due to browser limitation of getting content from system clipboard, paste using API and context menu option doesn't work. Hence, the paste option is disabled in context menu.

Alternatively, you can use the keyboard shortcuts,

- Cut: Ctrl + X
- Copy: Ctrl + C
- Paste: Ctrl + V

EnableLocalPaste behaviour

|EnableLocalPaste |Paste behavior details|

```
|-----|-----|
```

|True |Allows to paste content that is copied from the same Document Editor component alone and prevents pasting content from system clipboard. Hence the content copied from outside Document Editor component can't be pasted.
Browser limitation of pasting from system clipboard using API and context menu options, will be resolved. So, you can copy and paste content within the Document Editor component using API and context menu options too. |

|False |Allows to paste content from system clipboard. Hence the content copied from both the Document Editor component and outside can be pasted.
Browser limitation of pasting from system clipboard using API and context menu options, will remain as a limitation. |

Note:

- Keyboard shortcut for pasting will work properly in both cases.
- Copying content from Document Editor component and pasting outside will work properly in both cases.

Paste with formatting

Document Editor provides support to paste the system clipboard data with formatting. To enable clipboard paste with formatting options and copy/paste content from outside of Document editor, set the `enableLocalPaste` property in Document Editor to false and use this .NET Standard library [Syncfusion.EJ2.WordEditor.AspNet.Core](#) by the web API service implementation. This library helps you to paste the system clipboard data with formatting.

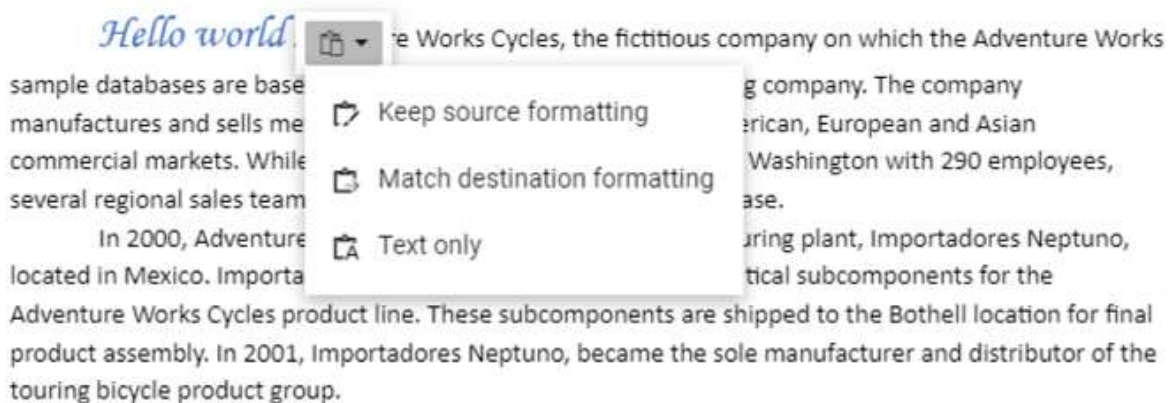
Refer this [page](#) for more details.

You can paste your system clipboard data in the following ways:

- **Keep Source Formatting** This option retains the character styles and direct formatting applied to the copied text. Direct formatting includes characteristics such as font size, italics, or other formatting that is not included in the paragraph style.
- **Match Destination Formatting** This option discards most of the formatting applied directly to the copied text, but it retains the formatting applied for emphasis, such as bold and italic when it is applied to only a portion of the selection. The text takes on the style characteristics of the paragraph where it is pasted. The text also takes on any direct formatting or character style properties of text that immediately precedes the cursor when the text is pasted.
- **Text Only** This option discards all formatting and non-text elements such as pictures or tables. The text takes on the style characteristics of the paragraph where it is pasted and takes on any direct formatting or character style properties of text that immediately precedes the cursor when the text is pasted. Graphical elements are discarded and tables are converted to a series of paragraphs.

This paste option appear as follows.

Adventure Works Cycles



See Also

- [Feature modules](#)
- [Keyboard shortcuts](#)

Collaborative Editing (preview)

Allows multiple users to work on the same document simultaneously. This can be done in real-time, so that collaborators can see the changes as they are made. Collaborative editing can be a great way to improve efficiency, as it allows team members to work together on a document without having to wait for others to finish their changes.

Note: Collaborative editing support is currently in preview mode only and is not yet ready for production environments.

Prerequisites

The following are needed to enable collaborative editing in Document Editor.

- SignalR
- Microsoft SQL Server

How to enable collaborative editing in client side

Step 1: Enable collaborative editing in Document Editor

To enable collaborative editing, inject `CollaborativeEditingHandler` and set the property `enableCollaborativeEditing` to true in the Document Editor, like in the code snippet below.

INDEX.JS

```
// Inject collaborative editing module.
ej.documenteditor.DocumentEditor.Inject(ej.documenteditor.CollaborativeEditingHandler);
ej.documenteditor.DocumentEditorContainer.Inject(ej.documenteditor.Toolbar);
let container = new ej.documenteditor.DocumentEditorContainer({
  enableToolbar: true, height: '590px', });
container.serviceUrl = 'http://localhost:5000/api/documenteditor/';
container.appendTo('#container');
// Enable collaborative editing in Document Editor.
container.documentEditor.enableCollaborativeEditing = true;
```

Step 2: Configure SignalR to send and receive changes

To broadcast the changes made and receive changes from remote users, configure SignalR like below.

INDEX.JS

```
let connectionId = "";
var connection = new signalR.HubConnectionBuilder().withUrl(serviceUrl +
  '/documenteditorhub', {
  skipNegotiation: true,
  transport: signalR.HttpTransportType.WebSockets
}).withAutomaticReconnect().build();
connection.onclose(async () => {
  if (connection.state === signalR.HubConnectionState.Disconnected) {
    alert('Connection lost. Please reload the browser to continue.');
```



```

        connection.send('JoinGroup', { roomName: data.roomName,
currentUser: data.currentUser });
        console.log('server connected!!!');
    });
} catch (err) {
    console.log(err);
    setTimeout(start, 5000);
}
};
// Event handler to handle data received
connection.on('dataReceived', onDataReceived.bind(this));
function onDataReceived(action, data) {
    if (connections) {
        if (action == 'connectionId') {
            connectionId = data;
        } else if (connectionId != data.connectionId) {
            if (action == 'action' || action == 'addUser') {
                // Add user info when new user join the collaborative
editing session.
                titleBar.addUser(data);
            } else if (action == 'removeUser') {
                // Remove user info from title bar once user is
disconnected.
                titleBar.removeUser(data);
            }
        }
        // Apply remote changes to current document.
        connections.applyRemoteAction(action, data);
    }
}
}

```

Step 3: Join SignalR room while opening the document

When opening a document, we need to generate a unique ID for each document. These unique IDs are then used to create rooms using SignalR, which facilitates sending and receiving data from the server.

INDEX.JS

```

function openDocument(responseText, roomName) {
    let data = JSON.parse(responseText);
    // Get collaborative editing module.
    connections =
container.documentEditor.collaborativeEditingHandlerModule;
    // Configure collaborative editing room name in collaborative editing
module
    connections.updateRoomInfo(roomName, data.version, serviceUrl);
    container.documentEditor.open(data.sfdt);
    setTimeout(function () {
        // connect to signalR room with specified name.
        start({ action: 'connect', roomName: roomName, currentUser:
container.currentUser }, null);
    });
}

```

Step 4: Broadcast current editing changes to remote users

Changes made on the client-side need to be sent to the server-side to broadcast them to other connected users. To send the changes made to the server, use the method shown below from the document editor using the `contentChange` event.

INDEX.JS

```
container.contentChange = function (args) {  
    if (connections) {  
        // Send current changes to server to broadcast it to other users.  
        connections.sendActionToServer(args.operations)  
    }  
}
```

How to enable collaborative editing in ASP.NET Core

Step 1: Configure SignalR in ASP.NET Core

We are using Microsoft SignalR to broadcast the changes. Please add the following configuration to your application's `Program.cs` file.

```
`csharp  
using Microsoft.Azure.SignalR;  
  
.....  
  
builder.Services.AddSignalR();  
  
.....  
  
.....  
  
.....  
  
app.MapHub<DocumentEditorHub>("/documenteditorhub");  
  
.....  
  
.....  
  
`
```

Step 2: Configure SignalR hub to create room for collaborative editing session

To manage groups for each document, create a folder named "Hub" and add a file named `DocumentEditorHub.cs` inside it. Add the following code to the file to manage SignalR groups using room names.

Join the group by using unique id of the document by using `JoinGroup` method.

```
`csharp  
  
static Dictionary<string, ActionInfo> userManager = new Dictionary<string, ActionInfo>();  
  
internal static Dictionary<string, List<ActionInfo>> groupManager = new Dictionary<string,  
List<ActionInfo>>();  
  
// Join to the specified room name  
  
public async Task JoinGroup(ActionInfo info)
```

```
{
if (!userManager.ContainsKey(Context.ConnectionId))
{
userManager.Add(Context.ConnectionId, info);
}
info.ConnectionId = Context.ConnectionId;
//Add the current connected use to the specified group
await Groups.AddToGroupAsync(Context.ConnectionId, info.RoomName);
if (groupManager.ContainsKey(info.RoomName))
{
await Clients.Caller.SendAsync("dataReceived", "addUser", groupManager[info.RoomName]);
}
lock (groupManager)
{
if (groupManager.ContainsKey(info.RoomName))
{
groupManager[info.RoomName].Add(info);
}
else
{
List<ActionInfo> actions = new List<ActionInfo>
{
info
};
groupManager.Add(info.RoomName, actions);
}
}
// Notify other users in the group about new user joined the collaborative editing session.
Clients.GroupExcept(info.RoomName, Context.ConnectionId).SendAsync("dataReceived", "addUser",
info);
}
```

Handle user disconnection using SignalR.

```
`csharp
//Handle disconnection from group.
public override Task OnDisconnectedAsync(Exception? e)
{
    string roomName = userManager[Context.ConnectionId].RoomName;
    if (groupManager.ContainsKey(roomName))
    {
        groupManager[roomName].Remove(userManager[Context.ConnectionId]);
        if (groupManager[roomName].Count == 0)
        {
            groupManager.Remove(roomName);
            //If all user disconnected from current room. Auto save the change to source document.
            CollaborativeEditingController.UpdateOperationsToSourceDocument(roomName,
            "<<documentpath>>", false);
        }
    }
    if (userManager.ContainsKey(Context.ConnectionId))
    {
        //Notify other user in the group about user exit the collaborative editing session
        Clients.OthersInGroup(roomName).SendAsync("dataReceived", "removeUser", Context.ConnectionId);
        Groups.RemoveFromGroupAsync(Context.ConnectionId, roomName);
        userManager.Remove(Context.ConnectionId);
    }
    return base.OnDisconnectedAsync(e);
}
`
```

Step 3: Configure Microsoft SQL database connection string in application level

Configure the SQL database that stores temporary data for the collaborative editing session. Provide the SQL database connection string in `appsettings.json` file.

```
`json
.....
"ConnectionStrings": {
    "DocumentEditorDatabase": "<SQL server connection string>"
}
```

.....
 、

Step 4: Configure Web API actions for collaborative editing

Import File

- When opening a document, create a database table to store temporary data for the collaborative editing session.
- If the table already exists, retrieve the records from the table and apply them to the `WordDocument` instance using the `UpdateActions` method before converting it to the SFDT format.

```
`csharp
public string ImportFile([FromBody] FileInfo param)
{
    .....
    .....
    DocumentContent content = new DocumentContent();
    .....
    //Get source document from database/file system/blob storage
    WordDocument document = GetDocumentFromDatabase(param.fileName, param.documentOwner);
    .....
    //Get temporary records from database
    List<ActionInfo> actions = CreatedTable(param.fileName);
    if(actions!=null)
    {
        //Apply temporary data to the document
        document.UpdateActions(actions);
    }
    string json = Newtonsoft.Json.JsonConvert.SerializeObject(document);
    content.version = 0;
    content.sfdt = json;
    return Newtonsoft.Json.JsonConvert.SerializeObject(content);
}
、
```

Update editing records to database

Each edit operation made by the user is sent to the server and is pushed to the database. Each operation receives a version number after being inserted into the database.

After inserting the records to the server, the position of the current editing operation must be transformed against any previous editing operations not yet synced with the client using the `TransformOperation` method.

After performing the transformation, the current operation is broadcast to all connected users within the group.

```
`csharp
public async Task<ActionInfo> UpdateAction([FromBody] ActionInfo param)
{
    try
    {
        ActionInfo modifiedAction = AddOperationsToTable(param);
        //After transformation broadcast changes to all users in the group
        await _hubContext.Clients.Group(param.RoomName).SendAsync("dataReceived", "action",
            modifiedAction);
        return modifiedAction;
    }
    catch
    {
        return null;
    }
}

private ActionInfo AddOperationsToTable(ActionInfo action)
{
    int clientVersion = action.Version;
    string tableName = action.RoomName;

    .....

    .....

    .....

    .....

    List<ActionInfo> actions = GetOperationsQueue(tableName);
    foreach (ActionInfo info in actions)
    {
```

```

if (!info.IsTransformed)
{
    CollaborativeEditingHandler.TransformOperation(info, actions);
}
}
action = actions[actions.Count - 1];
action.Version = updateVersion;
//Return the transformed operation to broadcast it to other clients.
return action;
}
,

```

[Add Web API to get previous operation as a backup to get lost operations](#)

On the client side, messages send from server using SignalR may be received in a different order, or some operations may be missed due to network issues. In these cases, we need a backup method to retrieve missing records from the database.

Using the following method, we can retrieve all operations after the last successful client-synced version and return all missing operations to the requesting client.

```

`csharp
public async Task<ActionInfo> UpdateAction([FromBody] ActionInfo param)
{
    try
    {
        ActionInfo modifiedAction = AddOperationsToTable(param);
        //After transformation broadcast changes to all users in the group
        await _hubContext.Clients.Group(param.RoomName).SendAsync("dataReceived", "action",
            modifiedAction);
        return modifiedAction;
    }
    catch
    {
        return null;
    }
}

private ActionInfo AddOperationsToTable(ActionInfo action)
{

```

```

int clientVersion = action.Version;
string tableName = action.RoomName;
.....
.....
.....
.....
List<ActionInfo> actions = GetOperationsQueue(table);
foreach (ActionInfo info in actions)
{
    if (!info.IsTransformed)
    {
        CollaborativeEditingHandler.TransformOperation(info, actions);
    }
}
action = actions[actions.Count - 1];
action.Version = updateVersion;
//Return the transformed operation to broadcast it to other clients.
return action;
}
`

```

Full version of the code discussed about can be found in below GitHub location.

GitHub Example: [Collaborative editing examples](#)

History in ##Platform_Name## Document editor control

Document Editor tracks the history of all editing actions done in the document, which allows undo and redo functionality.

Enable or disable history

Inject the 'EditorHistory' module in your application to provide history preservation functionality for 'DocumentEditor'. Refer to the following code example.

```

`ts
//Inject require modules.
DocumentEditor.Inject(Editor, Selection, EditorHistory);
let editor: DocumentEditor = new DocumentEditor({ enableEditor: true, isReadOnly: false });
//Enable editor history module.
editor.enableEditorHistory = true;

```


,

You can enable or disable history preservation for a Document Editor instance any time using the 'enableEditorHistory' property. Refer to the following sample code.

```
`ts
editor.enableEditorHistory = false;
```

,

Undo and redo

You can perform undo and redo by 'CTRL+Z' and 'CTRL+Y' keyboard shortcuts. Document Editor exposes API to do it programmatically.

To undo the last editing operation in Document Editor, refer to the following sample code.

```
`ts
editor.editorHistory.undo();
```

,

To redo the last undone action, refer to the following code example.

```
`ts
editor.editorHistory.redo();
```

,

Stack size

History of editing actions will be maintained in stack, so that the last item will be reverted first. By default, Document Editor limits the size of undo and redo stacks to 500 each respectively. However, you can customize this limit. Refer to the following sample code.

```
`ts
//Set undo limit.
editor.editorHistory.undoLimit = 400;
//Set redo limit.
editor.editorHistory.redoLimit = 400;
```

,

See Also

- [Feature modules](#)
- [Keyboard shortcuts](#)

Find and replace in `##Platform_Name##` Document editor control

The Document Editor component searches a portion of text in the document through a built-in interface called `OptionsPane` or rich APIs. When used in combination with selection performs various operations on the search results like replacing it with some other text, highlighting it, making it bolder, and more.

Options pane

This provides the options to search for a portion of text in the document. After search operation is completed, the search results will be displayed in a list and options to navigate between them. The current occurrence of matched text or all occurrences with another text can be replaced by switching to **Replace** tab. This pane is opened using the keyboard shortcut **CTRL+F**. You can also open it programmatically using the following sample code.

INDEX.JS

```
var documenteditor = new ej.documenteditor.DocumentEditor({ enableSelection:
true, enableSearch: true, enableEditor: true, isReadOnly: false,
enableOptionsPane: true });
documenteditor.appendTo('#DocumentEditor');
var sfdt = `{
  "sections": [
    {
      "blocks": [
        {
          "inlines": [
            {
              "characterFormat": {
                "bold": true,
                "italic": true
              },
              "text": "Adventure Works Cycles, the fictitious
company on which the AdventureWorks sample databases are based, is a large,
multinational manufacturing company. The company manufactures and sells
metal and composite bicycles to North American, European and Asian
commercial markets. While its base operation is located in Bothell,
Washington with 290 employees, several regional sales teams are located
throughout their market base."
            }
          ]
        }
      ]
    }
  ]
};
documenteditor.open(sfdt);
document.getElementById('showhidepane').addEventListener('click', () => {
  documenteditor.showOptionsPane();
});
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
documenteditor/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/fabric.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div style="display: inline">
            <button id="showhidepane" class="e-control e-btn e-
primary">Optionspane</button>
        </div>
        <div style="width:100%;height: 100%">
            <!--Element which will render as DocumentEditor -->
            <div id="DocumentEditor"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

You can close the options pane by pressing **Esc** key.

Search

The [Search](#) module of Document Editor exposes the following APIs:

API Name	Type	Description
---	---	---
findAll()	Method	Searches for specified text in the whole document and highlights it with yellow.
searchResults	Property	This is an instance of SearchResults .
find()	Method	Find immediate occurrence of specified text from cursor position in the document and highlights it with yellow.

Find the immediate occurrence in the document

Using [find\(\)](#) method, you can find the immediate occurrence of specified text from current cursor position in the document.

The following example code illustrates how to use find in Document editor.

```
`ts
documenteditor.search.find('Some text', 'None');
`
```

Note: Second parameter is optional parameter and it denotes find Options. Possible values of find options are `'None'` | `'WholeWord'` | `'CaseSensitive'` | `'CaseSensitiveWholeWord'`.

Find all the occurrences in the document

Using [findAll\(\)](#) method, you can find all the occurrences of specified text in the whole document and highlight it with yellow.

The following example code illustrates how to find All the text in the document.

```
`ts
documenteditor.search.findAll('Some text', 'None');
`
```

Note: Second parameter is optional parameter and it denotes find Options. Possible values of find options are `'None'` | `'WholeWord'` | `'CaseSensitive'` | `'CaseSensitiveWholeWord'`.

Search results

The [SearchResults](#) class provides information about the search results after a search operation is completed that can be identified using the [searchResultsChange](#) event. This will expose the following APIs:

API Name	Type	Description
---	---	---
length	Property	Returns the total number of results found on the search.
index	Property	Returns the index of selected search result. You can change the value for this property to move the selection.
replaceAll()	Method	Replaces all the occurrences with specified text.
clear()	Method	Clears the search result.

Replace all the occurrences

Using [replaceAll\(\)](#), you can replace all the occurrences with specified text.

The following example code illustrates how to use replace All in Document editor.

```
`ts
documentEditor.search.findAll('Some text');
// Replace all the searched text with word 'Mike'
documentEditor.search.searchResults.replaceAll("Mike");
`
```

Replace

Using [insertText](#), you can replace the current searched text with specified text and it replace single occurrence.

Note: This [insertText](#) API accepts following control characters

- * New line characters ("\\r", "\\r\\n", "\\n") - Inserts a new paragraph and appends the remaining text to the new paragraph.
- * Line break character ("\\v") - Moves the remaining text to start in new line.
- * Tab character ("\\t") - Allocates a tab space and continue the next character.

The following example code illustrates how to find a text in the document and replace each occurrence of the text one by one programmatically.

```
`ts
container.documentEditor.search.findAll('works');
let searchLength: number = container.documentEditor.search.searchResults.length;
for (let i = 0; i < searchLength; i++) {
// It will move selection to specific searched index,move to each occurrence one by one
container.documentEditor.search.searchResults.index = i;
// Replace it with some text
container.documentEditor.editor.insertText('Hello');
}
container.documentEditor.search.searchResults.clear();
```

SearchResultsChange event

[DocumentEditor](#) exposes the [searchResultsChange](#) event that will be triggered whenever search results are changed. Consider the following scenarios:

- A search operation is completed with some results.
- The results are replaced with some other text, since it will be cleared automatically.
- The results are cleared explicitly.

Refer to the following code example.

```
`ts
documenteditor.searchResultsChange = function() {
};
```

Customize find and replace

Using the exposed APIs, you can customize the find and replace functionality in your application. Refer to the following sample code.

INDEX.JS

```
var documenteditor = new ej.documenteditor.DocumentEditor({ enableSelection:
true, enableSearch: true, enableEditor: true, isReadOnly: false });
documenteditor.appendTo('#DocumentEditor');
var sfdt = `{
  "sections": [
    {
      "blocks": [
        {
          "inlines": [
            {
              "characterFormat": {
                "bold": true,
                "italic": true
              },
              "text": "Adventure Works Cycles, the fictitious
company on which the AdventureWorks sample databases are based, is a large,
multinational manufacturing company. The company manufactures and sells
metal and composite bicycles to North American, European and Asian
commercial markets. While its base operation is located in Bothell,
Washington with 290 employees, several regional sales teams are located
throughout their market base."
            }
          ]
        }
      ]
    }
  ]
};
documenteditor.open(sfdt);
document.getElementById('replace_all').addEventListener('click',function ()
{
  var textToFind = (document.getElementById('find_text')).value;
  var textToReplace = (document.getElementById('replace_text')).value;
  if (textToFind !== '') {
    // Find all the occurrences of given text
    documenteditor.searchModule.findAll(textToFind);
    if (documenteditor.searchModule.searchResults.length > 0) {
      // Replace all the occurrences of given text
      documenteditor.searchModule.searchResults.replaceAll(textToReplace);
    }
  }
});
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```

<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
documenteditor/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <table>
                <tbody><tr>
                    <td>
                        <label>Text to find:</label>
                    </td>
                    <td>
                        <input type="text" id="find_text">
                    </td>
                </tr>
                <tr>
                    <td>
                        <label>Text to replace:</label>
                    </td>
                    <td>
                        <input type="text" id="replace_text">
                    </td>
                </tr>
                <tr>
                    <td colspan="2">
                        <button id="replace_all" v-
on:click="onReplaceButtonClick" style="float:right;margin-top: 10px;"
class="e-control e-btn">Replace All</button>
                    </td>
                </tr>
            </tbody></table>
        </div>
        <div style="width:100%;height: 100%">
            <!--Element which will render as DocumentEditor -->
            <div id="DocumentEditor"></div>
        </div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Keyboard shortcut in ##Platform_Name## Document editor control

Text formatting

The following table lists the default keyboard shortcuts in Document Editor for formatting text:

Key combination	Description
-----	-----
Ctrl + B	Toggles the bold property of selected text.
Ctrl + I	Toggles the italic property of selected text.
Ctrl + U	Toggles the underline property of selected text.
Ctrl + +	Toggles the subscript formatting of selected text.
Ctrl + Shift + +	Toggles the superscript formatting of selected contents.
Ctrl + }	Increases the actual font size of selected text by one point.
Ctrl + {	Decreases the actual font size of selected text by one point.

Paragraph formatting

The following table lists the default keyboard shortcuts for formatting the paragraph:

Key combination	Description
-----	-----
Ctrl + E	Selected paragraphs are center aligned.
Ctrl + J	Selected paragraphs are justified.
Ctrl + L	Selected paragraphs are left aligned.
Ctrl + R	Selected paragraphs are right aligned.
Ctrl + 1	Single line spacing is applied for selected paragraphs.
Ctrl + 5	1.5 line spacing is applied for selected paragraphs.
Ctrl + 2	Double spacing is applied for selected paragraphs.
Ctrl + 0	No spacing is applied before the selected paragraphs.
Ctrl + M	Increases the left indent of selected paragraphs by a factor of 36 points.
Ctrl + Shift + M	Decreases the left indent of selected paragraphs by a factor of 36 points.
Ctrl + *	Show/Hide the hidden characters like spaces, tab, paragraph marks, and breaks.

Clipboard

Key Combination	Description
-----	-----
Ctrl + C	Copies selected contents to the clipboard.
Ctrl + V	Pastes plain text content from the clipboard.
Ctrl + X	Moves selected content to the clipboard.

Keyboard shortcut to navigate around the document

Key Combination	Description
----- -----	
Left arrow	Moves the cursor position one character to the left.
Right arrow	Moves the cursor position one character to the right.
Down arrow	Moves the cursor position down one line.
Up arrow	Moves the cursor position up one line.
Ctrl + Left arrow	Moves the cursor position one word to the left.
Ctrl + Right arrow	Moves the cursor position one word to the right.
Ctrl + Up arrow	Moves the cursor position one paragraph up.
Ctrl + Down arrow	Moves the cursor position one paragraph down.
Tab (in table)	Moves the cursor position one cell to the right.
Shift + Tab (in table)	Moves the cursor position one cell to the left.
Home	Moves the cursor position to the start of a line.
End	Moves the cursor position to the end of a line.
Page up	Moves the cursor position one screen up.
Page down	Moves the cursor position one screen down.
Ctrl + Home	Moves the cursor position to the start of a document.
Ctrl + End	Moves the cursor position to the end of a document.

Keyboard shortcut to extend selection

Key Combination	Description
----- -----	
Shift + Left arrow	Extends selection one character to the left.
Shift + Right arrow	Extends selection one character to the right.
Shift + Down arrow	Extends selection one line downward.
Shift + Up arrow	Extends selection one line upward.
Shift + Home	Extends selection to the start of a line.
Shift + End	Extends Selection to the end of a line.
Ctrl + A	Extends selection to the entire document.
Ctrl + Shift + Left arrow	Extends selection one word to the left.
Ctrl + Shift + Right arrow	Extends selection one word to the right.
Ctrl + Shift + Down arrow	Extends selection to the end of a paragraph.
Ctrl + Shift + Up arrow	Extends selection to the start of a paragraph.
Ctrl + Shift + Home	Extends selection to the start of a document.

|Ctrl + Shift + End| Extends selection to the end of a document. |

Find and Replace

|Key Combination|Description|

|-----|-----|

|Ctrl + F| Opens options pane. |

|Ctrl + H| Opens replace tab in options pane. |

Create, Save and Print document

|Key Combination|Description|

|-----|-----|

|Ctrl + N| Opens empty document. |

|Ctrl + S| Saves the document in SFDt format. |

|Ctrl + P| Prints the document. |

Edit Operation

|Key Combination|Description|

|-----|-----|

|Backspace | Deletes one character to the left. |

|Delete | Deletes one character to the right. |

|Ctrl + Z | Undo last performed action. |

|Ctrl + Y | Redo last undo action. |

Insert special characters

|Key Combination|Description|

|-----|-----|

|Ctrl + Enter | Inserts page break. |

|Shift + Enter | Inserts line break. |

Dialog

|Key Combination|Description|

|-----|-----|

|Ctrl + F| Opens options pane. |

|Ctrl + D| Opens font dialog. |

|Ctrl + K| Opens hyperlink dialog. |

See Also

- [How to override the keyboard shortcuts](#)

Scrolling zooming in ##Platform_Name## Document editor control

The Document Editor renders the document as page by page. You can scroll through the pages by mouse wheel or touch interactions. You can also scroll through the page by using 'scrollToPage()' method of Document Editor instance. Refer to the following code example.

INDEX.JS

```
var documenteditor = new ej.documenteditor.DocumentEditor({
  isReadOnly: false
});
documenteditor.appendTo('#DocumentEditor');
onLoadDefault();
documenteditor.scrollToPage(2);
function onLoadDefault() {
  var defaultDocument = {
    "sections": [
      {
        "blocks": [
          {
            "paragraphFormat": {
              "styleName": "Normal"
            },
            "inlines": [
              {
                "text": "First page"
              }
            ]
          }
        ],
        "headersFooters": {},
      },
      {
        "blocks": [
          {
            "paragraphFormat": {
              "styleName": "Normal"
            },
            "inlines": [
              {
                "text": "Second page"
              }
            ]
          }
        ],
        "headersFooters": {},
      }
    ],
    "characterFormat": {},
    "paragraphFormat": {},
    "background": {
      "color": "#FFFFFF"
    },
    "styles": [
      {
        "type": "Paragraph",
        "name": "Normal",
```

```

        "next": "Normal"
    },
    {
        "type": "Character",
        "name": "Default Paragraph Font"
    }
]
}
documenteditor.open(JSON.stringify(defaultDocument));
documenteditor.focusIn();
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
documenteditor/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/fabric.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="DocumentEditor">
    </div>
    <div id="page-fit-type-div"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {

```

```

    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Calling this method brings the specified page into view but doesn't move selection. Hence this method will work by default. That is, it works even if selection is not enabled.

In case, if you wish to move the selection to any page in Document Editor and bring it into view, you can use 'goToPage()' method of selection instance. Refer to the following code example.

INDEX.JS

```

var documenteditor = new ej.documenteditor.DocumentEditor({
  isReadOnly: false, serviceUrl:
  'https://services.syncfusion.com/js/production/api/documenteditor/'
});
documenteditor.enableAllModules();
documenteditor.appendTo('#DocumentEditor');
onLoadDefaultDocument();
documenteditor.viewer.selection.goToPage(3);
function onLoadDefaultDocument() {
  var defaultDocument = {
    "sections": [
      {
        "blocks": [
          {
            "paragraphFormat": {
              "styleName": "Normal"
            },
            "inlines": [
              {
                "text": "First page"
              }
            ]
          }
        ],
        "headersFooters": {},
      },
      {
        "blocks": [
          {
            "paragraphFormat": {
              "styleName": "Normal"
            },
            "inlines": [
              {
                "text": "Second page"
              }
            ]
          }
        ],
        "headersFooters": {},
      },
    ],
  }
}

```

```

        "blocks": [
            {
                "paragraphFormat": {
                    "styleName": "Normal"
                },
                "inlines": [
                    {
                        "text": "Third page"
                    }
                ]
            }
        ],
        "headersFooters": {},
    }
],
"characterFormat": {},
"paragraphFormat": {},
"background": {
    "color": "#FFFFFF"
},
"styles": [
    {
        "type": "Paragraph",
        "name": "Normal",
        "next": "Normal"
    },
    {
        "type": "Character",
        "name": "Default Paragraph Font"
    }
]
}
documenteditor.open(JSON.stringify(defaultDocument));
documenteditor.focusIn();
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
documenteditor/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/fabric.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="DocumentEditor">
        </div>
        <div id="page-fit-type-div"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Zooming

You can scale the contents in Document Editor ranging from 10% to 500% of the actual size. You can achieve this using mouse or touch interactions. You can also use 'zoomFactor' property of Document Editor instance. The value can be specified in a range from 0.1 to 5. Refer to the following code example.

`ts

```

import { DocumentEditor } from '@syncfusion/ej2-documenteditor';

//Initialize the Document Editor module.

let documenteditor: DocumentEditor = new DocumentEditor({
isReadOnly: false, serviceUrl: 'https://services.syncfusion.com/js/production/api/documenteditor/'
});

// Enable all the built in modules.
documenteditor.enableAllModules();
documenteditor.appendTo('#DocumentEditor');

//set zoom factor.
documenteditor.zoomFactor = 3;

```

Page Fit Type

Apart from specifying the zoom factor as value, the Document Editor provides option to specify page fit options such as fit to full page or fit to page width. You can set this option using 'fitPage' method of Document Editor instance. Refer to the following code example.

```
`ts
import { DocumentEditor } from '@syncfusion/ej2-documenteditor';

//Initialize the Document Editor module.

let documenteditor: DocumentEditor = new DocumentEditor({
isReadOnly: false, serviceUrl: 'https://services.syncfusion.com/js/production/api/documenteditor/'
});

// Enable all the built in modules.
documenteditor.enableAllModules();

documenteditor.appendTo('#DocumentEditor');

//Set zoom factor to fit page width.
documenteditor.fitPage('FitPageWidth');
```

Zoom option using UI

The following code example shows how to provide zoom options in Document Editor.

INDEX.JS

```
var documenteditor = new ej.documenteditor.DocumentEditor({
  isReadOnly: false, serviceUrl:
  'https://services.syncfusion.com/js/production/api/documenteditor/'
});
documenteditor.enableAllModules();
documenteditor.appendTo('#DocumentEditor');
var statusBarDiv = document.getElementById('page-fit-type-div');
var startPage = 1;
var label = document.createElement('label');
label.styles = 'margin-top: 6px;margin-right: 2px';
label.textContent = 'Page ';
statusBarDiv.appendChild(label);
var pageNumberLabel = document.createElement('label');
pageNumberLabel.id = 'documenteditor_page_no';
pageNumberLabel.style= 'text-transform:capitalize;white-
space:pre;overflow:hidden;user-select:none;cursor:text;height:17px;max-
width:150px';
var editablePageNumber = document.createElement('div');
pageNumberLabel.id= 'editablePageNumber';
pageNumberLabel.style= 'border: 1px solid #F1F1F1;display: inline-
flex;height: 17px;padding: 0px 4px';
pageNumberLabel.className= 'single-line e-de-pagenu-ber-text';
editablePageNumber.appendChild(pageNumberLabel);
updatePageNumber();
```



```
statusBarDiv.appendChild(editablePageNumber);
editablePageNumber.setAttribute('title', 'The current page number in the
document. Click or tap to navigate specific page.');
```

```
var label1 = document.createElement('label');
label1.id = 'documenteditor_pagecount';
label1.styles = 'margin-left:2px;varter-spacing: 1.05px;';
label1.textContent = 'of';
statusBarDiv.appendChild(label1);
var pageCount = document.createElement('label');
pageCount.id = 'documenteditor_pagecount'
pageCount.style = 'margin-left:6px;varter-spacing: 1.05px;';
updatePageCount();
statusBarDiv.appendChild(pageCount);
var editorPageCount = undefined;
var zoom;
var zoomBtn = document.createElement('button');
zoomBtn.id= 'documenteditor-zoom';
zoomBtn.className = 'e-de-statusbar-zoom';
statusBarDiv.appendChild(zoomBtn);
zoomBtn.setAttribute('title', 'Zoom level. Click or tap to open the Zoom
options.');
```

```
var items = [
  {
    text: '200%',
  },
  {
    text: '175%',
  },
  {
    text: '150%',
  },
  {
    text: '125%',
  },
  {
    text: '100%',
  },
  {
    text: '75%',
  },
  {
    text: '50%',
  },
  {
    text: '25%',
  },
  {
    separator: true
  },
  {
    text: 'Fit one page'
  },
  {
    text: 'Fit page width',
  },
];
```

```

zoom = new ej.splitbuttons.DropDownButton({ content: '100%', items: items,
select: onZoom }, zoomBtn);
editablePageNumber.addEventListener('click',
updateDocumentEditorPageNumber);
editablePageNumber.addEventListener('keydown', onKeyDown);
editablePageNumber.addEventListener('blur', onBlur);
documenteditor.viewChange = function (e) {
    updatePageNumberOnViewChange(e);
};
documenteditor.contentChange = function () {
    //Set page count
    updatePageCount();
};
function updatePageNumberOnViewChange(args) {
    if (documenteditor.selection
        && documenteditor.selection.startPage >= args.startPage &&
documenteditor.selection.startPage <= args.endPage) {
        startPage = documenteditor.selection.startPage;
    } else {
        startPage = args.startPage;
    }
    updatePageNumber();
}
function onBlur() {
    if (editablePageNumber.textContent === '' ||
parseInt(editablePageNumber.textContent, 0) > editorPageCount) {
        updatePageNumber();
    }
    editablePageNumber.contentEditable = 'false';
}
function onKeyDown(e) {
    if (e.which === 13) {
        e.preventDefault();
        var pageNumber = parseInt(editablePageNumber.textContent, 0);
        if (pageNumber > editorPageCount) {
            updatePageNumber();
        } else {
            if (documenteditor.selection) {
documenteditor.selection.goToPage(parseInt(editablePageNumber.textContent,
0));
            } else {
                documenteditor.scrollToPage(parseInt(editablePageNumber.textContent,
0));
            }
            editablePageNumber.contentEditable = 'false';
            if (editablePageNumber.textContent === '') {
                updatePageNumber();
            }
        }
    }
    if (e.which > 64) {
        e.preventDefault();
    }
}
function onZoom(args) {
    setZoomValue(args.item.text);
}

```

```

    updateZoomContent();
}
function setZoomValue(text) {
    if (text.match('Fit one page')) {
        documenteditor.fitPage('FitOnePage');
    } else if (text.match('Fit page width')) {
        documenteditor.fitPage('FitPageWidth');
    } else {
        documenteditor.zoomFactor = parseInt(text, 0) / 100;
    }
}
function updateZoomContent() {
    zoom.content = Math.round(documenteditor.zoomFactor * 100) + '%';
}
function updatePageNumber() {
    pageNumberLabel.textContent = startPage.toString();
}
function updatePageCount() {
    editorPageCount = documenteditor.pageCount;
    pageCount.textContent = editorPageCount.toString();
}
function updateDocumentEditorPageNumber() {
    var editablePageNumber = document.getElementById('editablePageNumber');
    editablePageNumber.contentEditable = 'true';
    editablePageNumber.focus();
    window.getSelection().selectAllChildren(editablePageNumber);
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
documenteditor/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/fabric.css" rel="stylesheet">

```

```
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="DocumentEditor">
            </div>
        <div id="page-fit-type-div"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Print in ##Platform_Name## Document editor control

To print the document, use the [print](#) method from Document Editor instance.

Refer to the following example for showing a document and print it.

INDEX.JS

```
var documenteditor = new ej.documenteditor.DocumentEditor({
    enablePrint: true
});
documenteditor.appendTo('#DocumentEditor');
var sfdt = {
    "sections": [
        {
            "blocks": [
                {
                    "inlines": [
                        {
                            "characterFormat": {
                                "bold": true,
                                "italic": true
                            },
                            "text": "Hello World"
                        }
                    ]
                }
            ]
        },
        {
            "headersFooters": {
            }
        }
    ]
};
documenteditor.open(JSON.stringify(sfdt));
var printButton = new ej.buttons.Button();
```

```
printButton.appendTo('#print');
document.getElementById('print').addEventListener('click',function (){
    documenteditor.print();
});
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
documenteditor/styles/fabric.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <button id="print">Print</button>
    <div id="DocumentEditor">
      </div>
    <div id="DocumentEditor2"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Refer to the following example for creating a document and print it.

INDEX.JS

```
var documenteditor = new ej.documenteditor.DocumentEditor({
  isReadOnly: false,
  enablePrint: true,
  enableEditor: true,
  enableSelection: true,
  enableEditorHistory: true
});
documenteditor.appendTo('#DocumentEditor');
var printButton = new ej.buttons.Button();
printButton.appendTo('#print');
document.getElementById('print').addEventListener('click',function (){
    documenteditor.print();
});
```

```
});
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
documenteditor/styles/fabric.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <button id="print">Print</button>
    <div id="DocumentEditor">
    </div>
    <div id="DocumentEditor2"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Improve print quality

Document editor provides an option to improve the print quality using [printDevicePixelRatio](#) in Document editor settings. Document editor using canvas approach to render content. Then, canvas are converted to image and it process for print. Using printDevicePixelRatio API, you can increase the image quality based on your requirement.

The following example code illustrates how to improve the print quality in Document editor container.

```
`ts
```

```
let container: DocumentEditorContainer = new DocumentEditorContainer({ enableToolbar: true, height:
'590px',
documentEditorSettings: {
  printDevicePixelRatio: 2
}
```

```
});
DocumentEditorContainer.Inject(Toolbar);
container.serviceUrl = 'https://services.syncfusion.com/js/production/api/documenteditor/';
container.appendTo('#container');
`
```

Note: By default, printDevicePixelRatio value is 1

Print using window object

You can print the document in Document Editor by passing the window instance. This is useful to implement print in third party frameworks such as electron, where the window instance will not be available. Refer to the following example.

```
`ts
import { DocumentEditor, Print } from '@syncfusion/ej2-documenteditor';
DocumentEditor.Inject(Print);
let documenteditor: DocumentEditor = new DocumentEditor({
  enablePrint: true, height: '370px'
});
documenteditor.appendTo('#DocumentEditor');
documenteditor.print(window);
`
```

Page setup

Some of the print options cannot be configured using JavaScript. Refer to the following links to learn more about the browser page setup:

- [Chrome](#)
- [Firefox](#)

However, you can customize margins, paper, and layout options by modifying the section format properties using page setup dialog

```
`ts
import { DocumentEditor, Print, PageSetupDialog, Editor, Selection, EditorHistory } from
 '@syncfusion/ej2-documenteditor';
DocumentEditor.Inject(Print, PageSetupDialog, Editor, Selection, EditorHistory);
let documenteditor: DocumentEditor = new DocumentEditor({
  isReadOnly: false,
  enablePrint: true,
  enablePageSetupDialog: true,
  enableEditor: true,
`
```

```

enableSelection: true,
enableEditorHistory: true,
height: '370px'
});
documenteditor.appendTo('#DocumentEditor');
documenteditor.showPageSetupDialog();
`

```

By customizing margins, papers, and layouts, the layout of the document will be changed in Document Editor. To modify these options during print operation, serialize the document as SFDT using the [serialize](#) method in Document Editor instance and open the SFDT data in another instance of Document Editor in separate window.

The following example shows how to customize layout options only for printing.

INDEX.JS

```

var documenteditor1 = new ej.documenteditor.DocumentEditor({
  isReadOnly: false,
  enablePrint: true,
  enableEditor: true,
  enableSelection: true,
  enableEditorHistory: true,
  enableSfdtExport: true
});
documenteditor1.appendTo('#DocumentEditor');
var documenteditor2 = new ej.documenteditor.DocumentEditor({
  isReadOnly: false,
  enablePrint: true,
  enableEditor: true,
  enableSelection: true,
  enableEditorHistory: true,
  enableSfdtExport: true
});
documenteditor2.appendTo('#DocumentEditor2');
var printButton = new ej.buttons.Button();
printButton.appendTo('#print');
document.getElementById('print').addEventListener('click', function () {
  var sfdtData = documenteditor1.serialize();
  documenteditor2.open(sfdtData);
  //Set A5 paper size
  documenteditor2.selection.sectionFormat.pageWidth = 419.55;
  documenteditor2.selection.sectionFormat.pageHeight = 595.30;
  documenteditor2.print();
});

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">

```



```
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
documenteditor/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="print">Print</button>
        <div id="DocumentEditor">
        </div>
        <div id="DocumentEditor2"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

See Also

- [Feature modules](#)
- [Page Setup dialog](#)

Dialog in ##Platform_Name## Document editor control

Document Editor provides dialog support to major operations such as insert or edit hyperlink, formatting text, paragraph, style, list and table properties.

Font Dialog

Font dialog allows you to modify all text properties for selected contents at once such as bold, italic, underline, font size, font color, strikethrough, subscript and superscript.

Refer to the following example.

INDEX.JS

```
var documenteditor = new ej.documenteditor.DocumentEditor({
enableFontDialog: true, enableSelection: true, enableEditor: true,
isReadOnly: false, enableEditorHistory: true });
documenteditor.appendTo('#DocumentEditor');
document.getElementById('dialog').addEventListener('click', function () {
    documenteditor.showDialog('Font');
});
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
documenteditor/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/fabric.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="toolbar">
      <button id="dialog">Dialog</button>
    </div>
    <div style="width:100%;height: 100%">
      <!--Element which will render as DocumentEditor -->
      <div id="DocumentEditor"></div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Paragraph dialog

This dialog allows modifying the paragraph formatting for selection at once such as text alignment, indentation, and spacing.

To open this dialog, refer to the following example.

INDEX.JS

```
var documenteditor = new ej.documenteditor.DocumentEditor({
  enableParagraphDialog: true, enableSelection: true, enableEditor: true,
  isReadOnly: false, enableEditorHistory: true });
documenteditor.appendTo('#DocumentEditor');
document.getElementById('dialog').addEventListener('click', function () {
  documenteditor.showDialog('Paragraph');
});
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
documenteditor/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/fabric.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="toolbar">
      <button id="dialog">Dialog</button>
    </div>
```

```

        <div style="width:100%;height: 100%">
            <!--Element which will render as DocumentEditor -->
            <div id="DocumentEditor"></div>
        </div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Table dialog

This dialog allows creating and inserting a table at cursor position by specifying the required number of rows and columns.

To open this dialog, refer to the following example.

INDEX.JS

```

var documenteditor = new ej.documenteditor.DocumentEditor({
enableTableDialog: true, enableSelection: true, enableEditor: true,
isReadOnly: false, enableEditorHistory: true });
documenteditor.appendTo('#DocumentEditor');
document.getElementById('dialog').addEventListener('click', function () {
    documenteditor.showDialog('Table');
});

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
documenteditor/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/fabric.css" rel="stylesheet">

```

```
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="toolbar">
            <button id="dialog">Dialog</button>
        </div>
        <div style="width:100%;height: 100%">
            <!--Element which will render as DocumentEditor -->
            <div id="DocumentEditor"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Bookmark dialog

This dialog allows you to perform the following operations:

- View all bookmarks.
- Navigate to a bookmark.
- Create a bookmark at current selection.
- Delete an existing bookmark.

To open this dialog, refer to the following example.

INDEX.JS

```
var documenteditor = new ej.documenteditor.DocumentEditor({
enableBookmarkDialog: true, enableSelection: true, enableEditor: true,
isReadOnly: false, enableEditorHistory: true });
documenteditor.appendTo('#DocumentEditor');
document.getElementById('dialog').addEventListener('click', function () {
    documenteditor.showDialog('Bookmark');
});
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
documenteditor/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="toolbar">
            <button id="dialog">Dialog</button>
        </div>
        <div style="width:100%;height: 100%">
            <!--Element which will render as DocumentEditor -->
            <div id="DocumentEditor"></div>
        </div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Hyperlink dialog

This dialog allows editing or inserting a hyperlink at cursor position.

To open this dialog, refer to the following example.

INDEX.JS

```

var documenteditor = new ej.documenteditor.DocumentEditor({
  enableHyperlinkDialog: true, enableSelection: true, enableEditor: true,
  isReadOnly: false, enableEditorHistory: true });
documenteditor.appendTo('#DocumentEditor');
document.getElementById('dialog').addEventListener('click', function () {
  documenteditor.showDialog('Hyperlink');
});

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
documenteditor/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/fabric.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="toolbar">
      <button id="dialog">Dialog</button>
    </div>
    <div style="width:100%;height: 100%">
      <!--Element which will render as DocumentEditor -->
      <div id="DocumentEditor"></div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {

```

```

    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Table of contents dialog

This dialog allows creating and inserting table of contents at cursor position. If the table of contents already exists at cursor position, you can customize its properties.

To open this dialog, refer to the following example.

INDEX.JS

```

var documenteditor = new ej.documenteditor.DocumentEditor({
enableTableOfContentsDialog: true, enableSelection: true, enableEditor:
true, isReadOnly: false, enableEditorHistory: true });
documenteditor.appendTo('#DocumentEditor');
document.getElementById('dialog').addEventListener('click', function () {
    documenteditor.showDialog('TableOfContents');
});

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
documenteditor/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/fabric.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>

```



```

<body>

  <div id="container">
    <div id="toolbar">
      <button id="dialog">Dialog</button>
    </div>
    <div style="width:100%;height: 100%">
      <!--Element which will render as DocumentEditor -->
      <div id="DocumentEditor"></div>
    </div>
  </div>
</body>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Styles Dialog

This dialog allows managing the styles in a document. It will display all the styles in the document with options to modify the properties of the existing style or create new style with the help of 'Style dialog'. Refer to the following example.

INDEX.JS

```

var documenteditor = new ej.documenteditor.DocumentEditor({
  enableStyleDialog: true, enableStylesDialog: true, enableSelection: true,
  enableEditor: true, isReadOnly: false, enableEditorHistory: true });
documenteditor.appendTo('#DocumentEditor');
document.getElementById('dialog').addEventListener('click', function () {
  documenteditor.showDialog('Styles');
});

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
documenteditor/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/fabric.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="toolbar">
            <button id="dialog">Dialog</button>
        </div>
        <div style="width:100%;height: 100%">
            <!--Element which will render as DocumentEditor -->
            <div id="DocumentEditor"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Style dialog

You can directly use this dialog for modifying any existing style or add new style by providing the style name.

To open this dialog, refer to the following example.

INDEX.JS

```

var documenteditor = new ej.documenteditor.DocumentEditor({
enableStyleDialog: true, enableSelection: true, enableEditor: true,
isReadOnly: false, enableEditorHistory: true });
documenteditor.appendTo('#DocumentEditor');
document.getElementById('dialog').addEventListener('click', function () {
    documenteditor.showDialog('Style');
});

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">

```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
documenteditor/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="toolbar">
            <button id="dialog">Dialog</button>
        </div>
        <div style="width:100%;height: 100%">
            <!--Element which will render as DocumentEditor -->
            <div id="DocumentEditor"></div>
        </div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

List dialog

This dialog allows creating a new list or modifying existing lists in the document.

To open this dialog, refer to the following example.

INDEX.JS

```

var documenteditor = new ej.documenteditor.DocumentEditor({
enableListDialog: true, enableSelection: true, enableEditor: true,
isReadOnly: false, enableEditorHistory: true });
documenteditor.appendTo('#DocumentEditor');
document.getElementById('dialog').addEventListener('click', function () {
    documenteditor.showDialog('List');
});

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
documenteditor/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/fabric.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="toolbar">
            <button id="dialog">Dialog</button>
        </div>
        <div style="width:100%;height: 100%">
            <!--Element which will render as DocumentEditor -->
            <div id="DocumentEditor"></div>
        </div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {

```

```

    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Borders and shading dialog

This dialog allows customizing the border style, border width, and background color of the table or selected cells.

To open this dialog, refer to the following example.

INDEX.JS

```

var documenteditor = new ej.documenteditor.DocumentEditor({
enableBordersAndShadingDialog: true, enableSelection: true, enableEditor:
true, isReadOnly: false, enableEditorHistory: true });
documenteditor.appendTo('#DocumentEditor');
documenteditor.editor.insertTable(2,2);
document.getElementById('dialog').addEventListener('click', function () {
    documenteditor.showDialog('BordersAndShading');
});

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
documenteditor/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/fabric.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>

```

```

</head>
<body>

  <div id="container">
    <div id="toolbar">
      <button id="dialog">Dialog</button>
    </div>
    <div style="width:100%;height: 100%">
      <!--Element which will render as DocumentEditor -->
      <div id="DocumentEditor"></div>
    </div>
  </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Table options dialog

This dialog allows customizing the default cell margins and spacing between each cells of the selected table.

To open this dialog, refer to the following example.

INDEX.JS

```

var documenteditor = new ej.documenteditor.DocumentEditor({
enableTableOptionsDialog: true, enableSelection: true, enableEditor: true,
isReadOnly: false, enableEditorHistory: true });
documenteditor.appendTo('#DocumentEditor');
documenteditor.editor.insertTable(2,2);
document.getElementById('dialog').addEventListener('click', function () {
  documenteditor.showDialog('TableOptions');
});

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
documenteditor/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/fabric.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="toolbar">
            <button id="dialog">Dialog</button>
        </div>
        <div style="width:100%;height: 100%">
            <!--Element which will render as DocumentEditor -->
            <div id="DocumentEditor"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Table properties dialog

This dialog allows customizing the table, row, and cell properties of the selected table.

To open this dialog, refer to the following example.

INDEX.JS

```

var documenteditor = new ej.documenteditor.DocumentEditor({
    isReadOnly: false,
    enableSelection: true,
    enableEditor: true,
    enableTableOptionsDialog: true,
    enableTablePropertiesDialog:true,
    enableBordersAndShadingDialog:true,
    enableSfdtExport: true,
});
var button = document.getElementById('dialog');
button.addEventListener('click', function() {
    //To open table properties dialog

```

```
documenteditor.showDialog('TableProperties');
});
documenteditor.appendTo('#DocumentEditor');
documenteditor.editor.insertTable(2,2);
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
documenteditor/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/fabric.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="toolbar">
      <button id="dialog">Dialog</button>
    </div>
    <div style="width:100%;height: 100%">
      <!--Element which will render as DocumentEditor -->
      <div id="DocumentEditor"></div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
```



```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Page setup dialog

This dialog allows customizing margins, size, and layout options for pages of the section.

To open this dialog, refer to the following example.

INDEX.JS

```
var documenteditor = new ej.documenteditor.DocumentEditor({
  enablePageSetupDialog: true, enableSelection: true, enableEditor: true,
  isReadOnly: false, enableEditorHistory: true });
documenteditor.appendTo('#DocumentEditor');
document.getElementById('dialog').addEventListener('click', function () {
  documenteditor.showDialog('PageSetup');
});
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
documenteditor/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/fabric.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="toolbar">
```

```

        <button id="dialog">Dialog</button>
    </div>
    <div style="width:100%;height: 100%">
        <!--Element which will render as DocumentEditor -->
        <div id="DocumentEditor"></div>
    </div>
</div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Column dialog

This dialog allows the end user to customize the number of columns, column width, and space between columns for the pages in a section.

To open this dialog, refer to the following example.

INDEX.JS

```

var documenteditor = new ej.documenteditor.DocumentEditor({
enablePageSetupDialog: true, enableSelection: true, enableEditor: true,
isReadOnly: false, enableEditorHistory: true });
documenteditor.appendTo('#DocumentEditor');
document.getElementById('dialog').addEventListener('click', function () {
    documenteditor.showDialog('PageSetup');
});

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
documenteditor/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="toolbar">
            <button id="dialog">Dialog</button>
        </div>
        <div style="width:100%;height: 100%">
            <!--Element which will render as DocumentEditor -->
            <div id="DocumentEditor"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Feature module](#)

R t l in ##Platform_Name## Document editor control

Document Editor provides RTL (right-to-left) support. This can be enabled using the “enableRtl” property.

INDEX.JS

```

ej.base.L10n.load({
  'ar-AE': {
    'documenteditor': {
      'Table': 'لجدول',
      'Row': 'الصف',
      'Cell': 'الخلية',
      'Ok': 'موافق',
      'Cancel': 'إلغاء الأمر',
      'Size': 'حجم',
      'Preferred Width': 'العرض المفضل',
      'Points': 'نقاط',
      'Percent': 'المائه',
      'Measure in': 'القياس في',
      'Alignment': 'محاذاة',
    }
  }
});

```

```

'Left': 'ليسار',
'Center': 'مركز',
'Right': 'الحق',
'Justify': 'تبرير',
'Indent from left': 'مسافة بادئه من اليسار',
'Borders and Shading': 'الحدود والتظليل',
'Options': 'خيارات',
'Specify height': 'تحديد الارتفاع',
'At least': 'الاقل',
'Exactly': 'تماما',
'Row height is': 'ارتفاع الصف هو',
'Allow row to break across pages': 'السماح بفصل الصف عبر',
الصفحات',
'Repeat as header row at the top of each page': 'تكرار كصف راس',
في اعلي كل صفحه',
'Vertical alignment': 'محاذاة عمودي',
'Top': 'أعلى',
'Bottom': 'اسفل',
'Default cell margins': 'هوامش الخلية الافتراضية',
'Default cell spacing': 'تباعد الخلايا الافتراضي',
'Allow spacing between cells': 'السماح بالتباعد بين الخلايا',
'Cell margins': 'هوامش الخلية',
'Same as the whole table': 'نفس الجدول بأكمله',
'Borders': 'الحدود',
'None': 'اي',
'Single': 'واحد',
'Dot': 'نقطه',
'DashSmallGap': 'داشسمجاب',
'DashLargeGap': 'الاتحاد الخاص',
'DashDot': 'داشدوت',
'DashDotDot': 'ددهدودوت',
'Double': 'انقر نقرا مزدوجا',
'Triple': 'الثلاثي',
'ThinThickSmallGap': 'فجوه صغيره سميكه رقيق',
'ThickThinSmallGap': 'الفجوة الصغيرة رقيقه سميكه',
'ThinThickThinSmallGap': 'فجوة صغيره سميكه رقيقه الفجوة الصغيرة',
'ThinThickMediumGap': 'فجوه متوسطه سميك',
'ThickThinMediumGap': 'سميكه الفجوة متوسطه رقيقه',
'ThinThickThinMediumGap': 'رقيقه سميكه متوسطه الفجوة',
'ThinThickLargeGap': 'الفجوة الكبيرة رقيقه سميكه',
'ThickThinLargeGap': 'فجوه كبيره رقيقه سميك',
'ThinThickThinLargeGap': 'رقيقه سميكه الفجوة الكبيرة',
'SingleWavy': 'واحد مائج',
'DoubleWavy': 'مزدوج مائج',
'DashDotStroked': 'اندفاعه نقطه القوية',
'Emboss3D': 'مزخرفD3',
'Engrave3D': 'نقشD3',
'Outset': 'البدايه',
'Inset': 'الداخلي',
'Thick': 'سميكه',
'Style': 'نمط',
'Width': 'عرض',
'Height': 'ارتفاع',
'Letter': 'رساله',
'Tabloid': 'التابلويد',
'Legal': 'القانونيه',
'Statement': 'بيان',

```

```

'Executive': 'التنفيذي',
'A3': 'A3',
'A4': 'A4',
'A5': 'A5',
'B4': 'B4',
'B5': 'B5',
'Custom Size': 'حجم مخصص',
'Different odd and even': 'مختلفه غريبه وحتى',
'Different first page': 'الصفحة الاولى مختلفه',
'From edge': 'من الحافة',
'Header': 'رأس',
'Footer': 'تذييل الصفحة',
'Margin': 'الهوامش',
'Paper': 'الورق',
'Layout': 'تخطيط',
'Orientation': 'التوجه',
'Landscape': 'المناظر الطبيعيه',
'Portrait': 'صوره',
'Table Of Contents': 'جدول المحتويات',
'Show page numbers': 'إظهار أرقام الصفحات',
'Right align page numbers': 'محاذاة أرقام الصفحات إلى اليمين',
'Nothing': 'شيء',
'Tab leader': 'قائد علامة التبويب',
'Show levels': 'إظهار المستويات',
'Use hyperlinks instead of page numbers': 'استخدام الارتباطات',
'التشعبية بدلا من أرقام الصفحات',
'Build table of contents from': 'بناء جدول محتويات من',
'Styles': 'انماط',
'Available styles': 'الأنماط المتوفرة',
'TOC level': 'مستوي جدول المحتويات',
'Heading': 'عنوان',
'Heading 1': 'عنوان 1',
'Heading 2': 'عنوان 2',
'Heading 3': 'عنوان 3',
'Heading 4': 'عنوان 4',
'Heading 5': 'عنوان 5',
'Heading 6': 'عنوان 6',
'List Paragraph': 'فقره القائمة',
'Normal': 'العادي',
'Outline levels': 'مستويات المخطط التفصيلي',
'Table entry fields': 'حقول إدخال الجدول',
'Modify': 'تعديل',
'Color': 'لون',
'Setting': 'اعداد',
'Box': 'مربع',
'All': 'جميع',
'Custom': 'المخصصه',
'Preview': 'معاينه',
'Shading': 'التظليل',
'Fill': 'ملء',
'Apply To': 'تنطبق علي',
'Table Properties': 'خصائص الجدول',
'Cell Options': 'خيارات الخلية',
'Table Options': 'خيارات الجدول',
'Insert Table': 'ادراج جدول',
'Number of columns': 'عدد الاعمده',
'Number of rows': 'عدد الصفوف',

```

```

'Text to display': 'النص الذي سيتم عرضه',
'Address': 'عنوان',
'Insert Hyperlink': 'ادراج ارتباط تشعبي',
'Edit Hyperlink': 'تحرير ارتباط تشعبي',
'Insert': 'ادراج',
'General': 'العامه',
'Indentation': 'المسافه البادئه',
'Before text': 'قبل النص',
'Special': 'الخاصه',
'First line': 'السطر الأول',
'Hanging': 'معلقه',
'After text': 'بعد النص',
'By': 'من',
'Before': 'قبل',
'Line Spacing': 'تباعد الأسطر',
'After': 'بعد',
'At': 'في',
'Multiple': 'متعدده',
'Spacing': 'تباعد',
'Define new Multilevel list': 'تحديد قائمه متعددة الاصعده جديده',
'List level': 'مستوي القائمة',
'Choose level to modify': 'اختر المستوي الذي تريد تعديله',
'Level': 'مستوي',
'Number format': 'تنسيق الأرقام',
'Number style for this level': 'نمط الرقم لهذا المستوي',
'Enter formatting for number': 'إدخال تنسيق لرقم',
'Start at': 'بداية من',
'Restart list after': 'أعاده تشغيل قائمه بعد',
'Position': 'موقف',
'Text indent at': 'المسافة البادئة للنص في',
'Aligned at': 'محاذاة في',
'Follow number with': 'اتبع الرقم مع',
'Tab character': 'حرف علامة التبويب',
'Space': 'الفضاء',
'Arabic': 'العربية',
'UpRoman': 'حتى الروماني',
'LowRoman': 'الرومانية منخفضه',
'UpLetter': '',
'LowLetter': '',
'Number': 'عدد',
'Leading zero': 'يؤدي صفر',
'Bullet': 'رصاصه',
'Ordinal': 'الترتيبيه',
'Ordinal Text': 'النص الترتيبي',
'For East': 'للشرق',
'No Restart': 'لا أعاده تشغيل',
'Font': 'الخط',
'Font style': 'نمط الخط',
'Underline style': 'نمط التسطير',
'Font color': 'لون الخط',
'Effects': 'الاثار',
'Strikethrough': 'يتوسطه',
'Superscript': 'مرتفع',
'Subscript': 'منخفض',
'Double strikethrough': 'خط مزدوج يتوسطه خط',
'Regular': 'العادي',
'Bold': 'جريئه',

```

```

'Italic': 'مائل',
'Cut': 'قطع',
'Copy': 'نسخ',
'Paste': 'لصق',
'Hyperlink': 'الارتباط التشعبي',
'Open Hyperlink': 'فتح ارتباط تشعبي',
'Copy Hyperlink': 'نسخ ارتباط تشعبي',
'Remove Hyperlink': 'أزاله ارتباط تشعبي',
'Paragraph': 'الفقره',
'Linked(Paragraph and Character)': 'مرتبط (فقره وحرف)',
'Character': 'حرف',
'Merge Cells': 'دمج الخلايا',
'Insert Above': 'ادراج أعلاه',
'Insert Below': 'ادراج أدناه',
'Insert Left': 'ادراج إلى اليسار',
'Insert Right': 'ادراج اليمين',
>Delete': 'حذف',
>Delete Table': 'حذف جدول',
>Delete Row': 'حذف صف',
>Delete Column': 'حذف عمود',
'File Name': 'اسم الملف',
'Format Type': 'نوع التنسيق',
'Save': 'حفظ',
'Navigation': 'التنقل',
'Results': 'نتائج',
'Replace': 'استبدال',
'Replace All': 'استبدال الكل',
'We replaced all': 'استبدلنا جميع',
'Find': 'العثور',
'No matches': 'لا توجد تطابقات',
'All Done': 'كل القيام به',
'Result': 'نتيجه',
'of': 'من',
'instances': 'الحالات',
'with': 'مع',
'Click to follow link': 'انقر لمتابعه الارتباط',
'Continue Numbering': 'متابعه الترقيم',
'Bookmark name': 'اسم الإشارة المرجعية',
'Close': 'اغلق',
'Restart At': 'أعاده التشغيل عند',
'Properties': 'خصائص',
'Name': 'اسم',
'Style type': 'نوع النمط',
'Style based on': 'نمط استنادا إلى',
'Style for following paragraph': 'نمط للفقره التاليه',
'Formatting': 'التنسيق',
'Numbering and Bullets': 'الترقيم والتعداد النقطي',
'Numbering': 'ترقيم',
'Update Field': 'تحديث الحقل',
'Edit Field': 'تحرير الحقل',
'Bookmark': 'الإشارة المرجعية',
'Page Setup': 'اعداد الصفحة',
'No bookmarks found': 'لم يتم العثور علي إشارات مرجعيه',
'Number format tooltip information': 'تنسيق رقم أحادي المستوي',
+ '</br>' + '[' + 'بأدئه' + '%[مستوي الاعداد] للاحقه' + '</br>'
// tslint:disable-next-line:max-line-length

```

```

+ 'على سبيل المثال ، "الفصل 1%". سيتم عرض الترقيم مثل' +
'</br>' + 'البند - الفصل الأول - البند' + '</br>' + 'الفصل الثاني - البند' + '</br>...' +
'</br>' + 'الفصل نون-البند' + '</br>'
// tslint:disable-next-line:max-line-length
+ '</br>' + 'تنسيق رقم متعدد الإعدادات' + '</br>' +
'[%مستوي المستوي]' + 'باده' + 'باده' + 'لاحقه' + 'لاحقه' +
'[%مستوي]' + 'لاحقه' +
+ 'على سبيل المثال ، "1% 2%". سيتم عرض الترقيم' +
'</br>' + 'البند 1.1' + '</br>' + 'البند 1.2' + '</br>...' + '</br>'
+ 'نون-البند 1.',
'Format': 'تنسيق',
'Create New Style': 'إنشاء نمط جديد',
'Modify Style': 'تعديل النمط',
'New': 'الجديد',
'Bullets': 'الرصص',
'Use bookmarks': 'استخدام الإشارات المرجعية',
'Table of Contents': 'جدول المحتويات',
'AutoFit': 'الاحتواء',
'AutoFit to Contents': 'احتواء تلقائي للمحتويات',
'AutoFit to Window': 'احتواء تلقائي للإطار',
'Fixed Column Width': 'عرض العمود الثابت',
'Reset': 'اعاده تعيين',
'Match case': 'حاله المباراة',
'Whole words': 'كلمات كامل',
'Add': 'اضافه',
'Go To': 'الانتقال إلى',
'Search for': 'البحث عن',
'Replace with': 'استبدال',
'TOC 1': 'جدول المحتويات 1',
'TOC 2': 'جدول المحتويات 2',
'TOC 3': 'جدول المحتويات 3',
'TOC 4': 'جدول المحتويات 4',
'TOC 5': 'جدول المحتويات 5',
'TOC 6': 'جدول المحتويات 6',
'TOC 7': 'جدول المحتويات 7',
'TOC 8': 'جدول المحتويات 8',
'TOC 9': 'جدول المحتويات 9',
'Right-to-left': 'من اليمين إلى اليسار',
'Left-to-right': 'من اليسار إلى اليمين',
'Direction': 'الاتجاه',
'Table direction': 'اتجاه الجدول',
'Indent from right': 'مسافة بادئه من اليمين',
'Page': 'صفحه',
'Fit one page': 'احتواء صفحه واحد',
'Fit page width': 'احتواء عرض الصفحة',
// tslint:disable-next-line:max-line-length
'The current page number in the document. Click or tap to
navigate specific page.': 'رقم الصفحة الحالية في المستند. انقر أو اضغط'
'للتنقل في صفحه معينه'
},
'colorpicker': {
'Apply': 'تطبيق',
'Cancel': 'إلغاء الأمر',
'ModeSwitcher': 'مفتاح كهربائي الوضع'
}
}
});

```



```

var documenteditor = new ej.documenteditor.DocumentEditor({isReadOnly:
false,enableRtl: true,locale: 'ar-AE', serviceUrl:
'https://services.syncfusion.com/js/production/api/documenteditor/'});
documenteditor.enableAllModules();
var containerPanel = document.getElementById('container');
function updateContainerSize() {
    this.containerPanel.style.height = window.innerHeight + 'px';
}
updateContainerSize();
documenteditor.appendTo('#DocumentEditor');
var sfdt=`{
  "sections": [
    {
      "blocks": [
        {
          "characterFormat": {
            "fontSize": 18.0,
            "fontFamily": "Calibri",
            "fontFamilyBidi": "Calibri"
          },
          "paragraphFormat": {
            "beforeSpacing": 18.0,
            "afterSpacing": 30.0,
            "styleName": "Heading 1",
            "bidi": true
          },
          "inlines": [
            {
              "text": "اعمال المغامرة دورات",
              "characterFormat": {
                "fontSize": 18.0,
                "bidi": true,
                "fontSizeBidi": 18.0
              }
            }
          ]
        }
      ]
    }
  ]
};
documenteditor.open(sfdt);

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
documenteditor/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/fabric.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="toolbar">
        </div>
        <div style="width:100%;height: 100%">
            <!--Element which will render as DocumentEditor -->
            <div id="DocumentEditor"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Chart in ##Platform_Name## Document editor control

Document Editor provides chart preservation support. Using Document Editor, you can see the chart reports from your Word document.

The following example shows chart preservation in Document Editor.

INDEX.JS

```

var documenteditor = new ej.documenteditor.DocumentEditor({ isReadOnly:
false, serviceUrl:
'https://services.syncfusion.com/js/production/api/documenteditor/' });
documenteditor.acceptTab = true;
documenteditor.enableAllModules();
documenteditor.pageOutline = '#E0E0E0';
documenteditor.appendTo('#DocumentEditor');

```

```

var sfdt
={
  "sections": [
    {
      "sectionFormat": {
        "pageWidth": 612, "pageHeight": 792, "leftMargin": 72, "rightMargin": 72, "topMargin": 72, "bottomMargin": 72, "differentFirstPage": false, "differentOddAndEvenPages": false, "headerDistance": 36, "footerDistance": 36, "bidi": false
      },
      "blocks": [
        {
          "paragraphFormat": {
            "textAlignment": "Center", "afterSpacing": 0, "lineSpacing": 1, "lineSpacingType": "Multiple", "styleName": "Normal", "listFormat": {}
          },
          "characterFormat": {
            "bold": true, "fontSize": 12, "fontFamily": "Verdana", "fontSizeBidi": 12, "fontFamilyBidi": "Verdana"
          },
          "inlines": [
            {
              "characterFormat": {
                "bold": true, "fontSize": 14, "fontFamily": "Verdana", "fontColor": "#17365DFF", "styleName": "a", "fontSizeBidi": 14, "fontFamilyBidi": "Verdana"
              },
              "text": "Northwind Management Report"
            }
          ]
        },
        {
          "paragraphFormat": {
            "afterSpacing": 0, "lineSpacing": 1, "lineSpacingType": "Multiple", "styleName": "Normal", "listFormat": {}
          },
          "characterFormat": {
            "fontSize": 10, "fontFamily": "Verdana", "fontSizeBidi": 10, "fontFamilyBidi": "Verdana"
          },
          "inlines": [
            {
              "paragraphFormat": {
                "afterSpacing": 0, "styleName": "Normal", "listFormat": {}
              },
              "characterFormat": {},
              "inlines": [
                {
                  "characterFormat": {
                    "fontSize": 10, "fontFamily": "Verdana", "styleName": "a", "fontSizeBidi": 10, "fontFamilyBidi": "Verdana"
                  },
                  "text": "This management report provides information obtained through data analysis, regarding the"
                },
                {
                  "characterFormat": {
                    "fontSize": 10, "fontFamily": "Verdana", "styleName": "a", "fontSizeBidi": 10, "fontFamilyBidi": "Verdana"
                  },
                  "text": "performance of Northwind Traders. This report will pay particular"
                },
                {
                  "characterFormat": {
                    "fontSize": 10, "fontFamily": "Verdana", "styleName": "a", "fontSizeBidi": 10, "fontFamilyBidi": "Verdana"
                  },
                  "text": "attention to the"
                },
                {
                  "characterFormat": {
                    "fontSize": 10, "fontFamily": "Verdana", "styleName": "a", "fontSizeBidi": 10, "fontFamilyBidi": "Verdana"
                  },
                  "text": "best-selling products, of our company."
                },
                {
                  "characterFormat": {
                    "fontSize": 10, "fontFamily": "Verdana", "fontSizeBidi": 10, "fontFamilyBidi": "Times New Roman"
                  },
                  "text": "The best-selling products of Northwind Traders"
                },
                {
                  "characterFormat": {
                    "fontSize": 10, "fontFamily": "Verdana", "fontSizeBidi": 10, "fontFamilyBidi": "Times New Roman"
                  },
                  "text": "Company as follows:"
                }
              ]
            }
          ]
        },
        {
          "paragraphFormat": {
            "afterSpacing": 0, "styleName": "Normal", "listFormat": {}
          },
          "characterFormat": {},
          "inlines": [
            {
              "rows": [
                {
                  "cells": [
                    {
                      "blocks": [
                        {
                          "paragraphFormat": {
                            "rightIndent": 26.850000381469727, "styleName": "Normal", "listFormat": {}
                          },
                          "characterFormat": {},
                          "inlines": [
                            {
                              "characterFormat": {
                                "fontSize": 10, "fontFamily": "Verdana", "fontSizeBidi": 10, "fontFamilyBidi": "Times New Roman"
                              },
                              "text": "S.No"
                            }
                          ]
                        }
                      ]
                    }
                  ]
                }
              ]
            }
          ]
        },
        {
          "cellFormat": {
            "borders": {
              "top": {
                "color": "#4472C4FF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0
              },
              "left": {
                "color": "#4472C4FF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0
              },
              "right": {
                "color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0
              },
              "bottom": {
                "color": "#4472C4FF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0
              },
              "diagonalDown": {
                "color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0
              },
              "diagonalUp": {
                "color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0
              },
              "horizontal": {
                "color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0
              },
              "vertical": {
                "color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0
              }
            },
            "shading": {
              "backgroundColor": "#4472C4FF", "foregroundColor": "empty", "textureStyle": "TextureNone"
            },
            "preferredWidth": 13.420000076293945, "preferredWidthType": "Percent", "cellWidth": 64.71214527422465, "columnSpan": 1, "rowSpan": 1, "verticalAlignment": "Top", "columnIndex": 0
          },
          "blocks": [
            {
              "paragraphFormat": {
                "styleName": "Normal", "listFormat": {}
              },
              "characterFormat": {},
              "inlines": [
                {
                  "characterFormat": {
                    "fontSize": 10, "fontFamily": "Verd

```

```

ana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"Product
Name"]]]], "cellFormat":{"borders":{"top":{"color":"#4472C4FF","hasNoneStyle":
:false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":
{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":
0.5,"shadow":false,"space":0},"right":{"color":"#8EAADBFF","hasNoneStyle":fa
lse,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":
{"color":"#4472C4FF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0
.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":
:false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonal
Up":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0
,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":fa
lse,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"c
olor":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shad
ow":false,"space":0},"shading":{"backgroundColor":"#4472C4FF","foregroundCo
lor":"empty","textureStyle":"TextureNone"},"preferredWidth":48.8600006103515
6,"preferredWidthType":"Percent","cellWidth":292.87942351880633,"columnSpan"
:1,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":1},{ "blocks":[{"para
graphFormat":{"styleName":"Normal","listFormat":{},"characterFormat":{},"in
lines":[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBi
di":10,"fontFamilyBidi":"Times New Roman"},"text":"Sum of Sales(in
$)"}]]], "cellFormat":{"borders":{"top":{"color":"#4472C4FF","hasNoneStyle":f
alse,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{
"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.
5,"shadow":false,"space":0},"right":{"color":"#4472C4FF","hasNoneStyle":fals
e,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"
color":"#4472C4FF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5
,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":
false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":
{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"
shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":fals
e,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"co
lor":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":
false,"space":0},"shading":{"backgroundColor":"#4472C4FF","foregroundColo
r":"empty","textureStyle":"TextureNone"},"preferredWidth":37.720001220703125
,"preferredWidthType":"Percent","cellWidth":117.95841899993776,"columnSpan":
1,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":2}], "rowFormat":{"hei
ght":14.399999618530273,"allowBreakAcrossPages":true,"heightType":"Exactly",
"isHeader":false,"borders":{"top":{"color":"#8EAADBFF","hasNoneStyle":false,
"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"colo
r":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"sh
adow":false,"space":0},"right":{"color":"#8EAADBFF","hasNoneStyle":false,"li
neStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":
"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"sha
adow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false
,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"c
olor":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shado
w":false,"space":0},"horizontal":{"color":"#8EAADBFF","hasNoneStyle":false,"
lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"vertical":{"c
olor":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,
"shadow":false,"space":0},"gridBefore":0,"gridBeforeWidth":0,"gridBeforeWid
thType":"Point","gridAfter":0,"gridAfterWidth":0,"gridAfterWidthType":"Point
"}},{ "cells":[{"blocks":[{"paragraphFormat":{"styleName":"Normal","listForma
t":{},"characterFormat":{},"inlines":[{"characterFormat":{"fontSize":10,"fo
ntFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New
Roman"},"text":"1"}]]], "cellFormat":{"borders":{"top":{"color":"#8EAADBFF","
hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"spa
ce":0},"left":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single"

```

```
"lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0}},{"shading":{"backgroundColor":"#D9E2F3FF"},"foregroundColor":"","textureStyle":"TextureNone"},"preferredWidth":13.420000076293945,"preferredWidthType":"Percent","cellWidth":64.71214527422465,"columnSpan":1,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":0},{ "blocks":[{"paragraphFormat":{"styleName":"Normal","listFormat":{}}, "characterFormat":{},"inlines":[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"Côte de Blaye"}]}],"cellFormat":{"borders":{"top":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0}}, {"shading":{"backgroundColor":"#D9E2F3FF"},"foregroundColor":"","textureStyle":"TextureNone"},"preferredWidth":48.86000061035156,"preferredWidthType":"Percent","cellWidth":292.87942351880633,"columnSpan":1,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":1},{ "blocks":[{"paragraphFormat":{"styleName":"Normal","listFormat":{}}, "characterFormat":{},"inlines":[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"141.396"}]}],"cellFormat":{"borders":{"top":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0}}, {"shading":{"backgroundColor":"#D9E2F3FF"},"foregroundColor":"","textureStyle":"TextureNone"},"preferredWidth":37.720001220703125,"preferredWidthType":"Percent","cellWidth":117.95841899993776,"columnSpan":1,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":2}], "rowFormat":{"height":14.399999618530273,"allowBreakAcrossPages":true,"heightType":"Exactly","isHeader":false,"borders":{"top":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0}}}
```

```

":0}, "bottom": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single",
"lineWidth": 0.5, "shadow": false, "space": 0}, "diagonalDown": {"color": "#000000",
"hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space":
0}, "diagonalUp": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None",
"lineWidth": 0, "shadow": false, "space": 0}, "horizontal": {"color": "#8EADBFF", "
hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "spa
ce": 0}, "vertical": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Sin
gle", "lineWidth": 0.5, "shadow": false, "space": 0}}, "gridBefore": 0, "gridBeforeWi
dth": 0, "gridBeforeWidthType": "Point", "gridAfter": 0, "gridAfterWidth": 0, "gridA
fterWidthType": "Point"}}, {"cells": [{"blocks": [{"paragraphFormat": {"styleName
": "Normal", "listFormat": {}, "characterFormat": {}, "inlines": [{"characterForma
t": {"fontSize": 10, "fontFamily": "Verdana", "fontSizeBidi": 10, "fontFamilyBidi":
"Times New
Roman"}, "text": "2"}]}]}, "cellFormat": {"borders": {"top": {"color": "#8EADBFF", "
hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "spa
ce": 0}, "left": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single",
"lineWidth": 0.5, "shadow": false, "space": 0}, "right": {"color": "#8EADBFF", "has
NoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space"
: 0}, "bottom": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single",
"lineWidth": 0.5, "shadow": false, "space": 0}, "diagonalDown": {"color": "#000000",
"hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space"
: 0}, "diagonalUp": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None",
"lineWidth": 0, "shadow": false, "space": 0}, "horizontal": {"color": "#000000", "has
NoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0},
"vertical": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineW
idth": 0, "shadow": false, "space": 0}}, "shading": {"backgroundColor": "#FFFFFFF",
"foregroundColor": "empty", "textureStyle": "TextureNone"}, "preferredWidth": 13.
420000076293945, "preferredWidthType": "Percent", "cellWidth": 64.71214527422465
, "columnSpan": 1, "rowSpan": 1, "verticalAlignment": "Top", "columnIndex": 0}, {"bl
ocks": [{"paragraphFormat": {"styleName": "Normal", "listFormat": {}}, "characterF
ormat": {}, "inlines": [{"characterFormat": {"fontSize": 10, "fontFamily": "Verdana
", "fontSizeBidi": 10, "fontFamilyBidi": "Times New Roman"}, "text": "Thüringer
Rostbratwurst"}]}]}, "cellFormat": {"borders": {"top": {"color": "#8EADBFF", "hasN
oneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space":
0}, "left": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "li
neWidth": 0.5, "shadow": false, "space": 0}, "right": {"color": "#8EADBFF", "hasNone
Style": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0},
"bottom": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lin
eWidth": 0.5, "shadow": false, "space": 0}, "diagonalDown": {"color": "#000000", "has
NoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0},
"diagonalUp": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lin
eWidth": 0, "shadow": false, "space": 0}, "horizontal": {"color": "#000000", "hasNone
Style": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "ver
tical": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth
": 0, "shadow": false, "space": 0}}, "shading": {"backgroundColor": "#FFFFFFF", "for
egroundColor": "empty", "textureStyle": "TextureNone"}, "preferredWidth": 48.8600
0061035156, "preferredWidthType": "Percent", "cellWidth": 292.87942351880633, "co
lumnSpan": 1, "rowSpan": 1, "verticalAlignment": "Top", "columnIndex": 1}, {"blocks
": [{"paragraphFormat": {"styleName": "Normal", "listFormat": {}}, "characterForma
t": {}, "inlines": [{"characterFormat": {"fontSize": 10, "fontFamily": "Verdana", "f
ontSizeBidi": 10, "fontFamilyBidi": "Times New
Roman"}, "text": "80.368"}]}]}, "cellFormat": {"borders": {"top": {"color": "#8EADB
FF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false
, "space": 0}, "left": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Si
ngle", "lineWidth": 0.5, "shadow": false, "space": 0}, "right": {"color": "#8EADBFF",
"hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "s
pace": 0}, "bottom": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Sin

```

```

gle", "lineWidth": 0.5, "shadow": false, "space": 0}, "diagonalDown": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "diagonalUp": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "horizontal": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "vertical": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}}, "shading": {"backgroundColor": "#FFFFFF", "foregroundColor": "empty", "textureStyle": "TextureNone"}, "preferredWidth": 37.720001220703125, "preferredWidthType": "Percent", "cellWidth": 117.95841899993776, "columnSpan": 1, "rowSpan": 1, "verticalAlignment": "Top", "columnIndex": 2}], "rowFormat": {"height": 14.399999618530273, "allowBreakAcrossPages": true, "rightType": "Exactly", "isHeader": false, "borders": {"top": {"color": "#8EAADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "left": {"color": "#8EAADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "right": {"color": "#8EAADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "bottom": {"color": "#8EAADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "diagonalDown": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "diagonalUp": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "horizontal": {"color": "#8EAADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "vertical": {"color": "#8EAADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}}, "gridBefore": 0, "gridBeforeWidth": 0, "gridBeforeWidthType": "Point", "gridAfter": 0, "gridAfterWidth": 0, "gridAfterWidthType": "Point"}}, {"cells": [{"blocks": [{"paragraphFormat": {"styleName": "Normal", "listFormat": {}, "characterFormat": {}, "inlines": [{"characterFormat": {"fontSize": 10, "fontFamily": "Verdana", "fontSizeBidi": 10, "fontFamilyBidi": "Times New Roman"}, "text": "3"}]}]}, "cellFormat": {"borders": {"top": {"color": "#8EAADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "left": {"color": "#8EAADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "right": {"color": "#8EAADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "bottom": {"color": "#8EAADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "diagonalDown": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "diagonalUp": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "horizontal": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "vertical": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}}, "shading": {"backgroundColor": "#D9E2F3FF", "foregroundColor": "empty", "textureStyle": "TextureNone"}, "preferredWidth": 13.420000076293945, "preferredWidthType": "Percent", "cellWidth": 64.71214527422465, "columnSpan": 1, "rowSpan": 1, "verticalAlignment": "Top", "columnIndex": 0}, {"blocks": [{"paragraphFormat": {"styleName": "Normal", "listFormat": {}, "characterFormat": {}, "inlines": [{"characterFormat": {"fontSize": 10, "fontFamily": "Verdana", "fontSizeBidi": 10, "fontFamilyBidi": "Times New Roman"}, "text": "Raclette Courdavault"}]}]}, "cellFormat": {"borders": {"top": {"color": "#8EAADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "left": {"color": "#8EAADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "right": {"color": "#8EAADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "bottom": {"color": "#8EAADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "diagonalDown": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "diagonalUp": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineW

```



```

idth":0,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0}},"shading":{"backgroundColor":"#D9E2F3FF","foregroundColor":"empty","textureStyle":"TextureNone"},"preferredWidth":48.86000061035156,"preferredWidthType":"Percent","cellWidth":292.87942351880633,"columnSpan":1,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":1},{
"blocks":
[{"paragraphFormat":{"styleName":"Normal","listFormat":{},"characterFormat":{},"inlines":[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"71.155"}]}]},"cellFormat":{"borders":{"top":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0}},"shading":{"backgroundColor":"#D9E2F3FF","foregroundColor":"empty","textureStyle":"TextureNone"},"preferredWidth":37.720001220703125,"preferredWidthType":"Percent","cellWidth":117.95841899993776,"columnSpan":1,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":2
}],
"rowFormat":{"height":14.399999618530273,"allowBreakAcrossPages":true,"heightType":"Exactly","isHeader":false,"borders":{"top":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"vertical":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0}},"gridBefore":0,"gridBeforeWidth":0,"gridBeforeWidthType":"Point","gridAfter":0,"gridAfterWidth":0,"gridAfterWidthType":"Point"}},{
"cells":
[{"blocks":
[{"paragraphFormat":{"styleName":"Normal","listFormat":{},"characterFormat":{},"inlines":[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"4"}]}]}]},"cellFormat":{"borders":{"top":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineW

```



```

idth":0,"shadow":false,"space":0}},{"shading":{"backgroundColor":"#FFFFFFF","
foregroundColor":"empty","textureStyle":"TextureNone"},"preferredWidth":13.
420000076293945,"preferredWidthType":"Percent","cellWidth":64.71214527422465
,"columnSpan":1,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":0},{
"blocks":[{"paragraphFormat":{"styleName":"Normal","listFormat":{}},{"characterF
ormat":{},"inlines":[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana
","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"Tarte au
sucr
e"}]}]},"cellFormat":{"borders":{"top":{"color":"#8EADBFF","hasNoneStyle":fal
se,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"c
olor":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,
"shadow":false,"space":0},"right":{"color":"#8EADBFF","hasNoneStyle":false,
"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"co
lor":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"
shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":fa
lse,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":
{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"sh
adow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":false,
"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"colo
r":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":
false,"space":0}},{"shading":{"backgroundColor":"#FFFFFFF","foregroundColor
":"empty","textureStyle":"TextureNone"},"preferredWidth":48.86000061035156,"p
referredWidthType":"Percent","cellWidth":292.87942351880633,"columnSpan":1,"
rowSpan":1,"verticalAlignment":"Top"},"columnIndex":1},{
"blocks":[{"paragrap
hFormat":{"styleName":"Normal","listFormat":{}},{"characterFormat":{},"inline
s":[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":
10,"fontFamilyBidi":"Times New
Roman"},"text":"47.234"}]}]},"cellFormat":{"borders":{"top":{"color":"#8EADBFF","hasNoneStyle":
false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":
{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0
.5,"shadow":false,"space":0},"right":{"color":"#8EADBFF","hasNoneStyle":fal
se,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{
"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.
5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle"
:false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalU
p":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,
"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":fal
se,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"c
olor":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shado
w":false,"space":0}},{"shading":{"backgroundColor":"#FFFFFFF","foregroundCol
or":"empty","textureStyle":"TextureNone"},"preferredWidth":37.72000122070312
5,"preferredWidthType":"Percent","cellWidth":117.95841899993776,"columnSpan"
:1,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":2}],{"rowFormat":{"he
ight":14.399999618530273,"allowBreakAcrossPages":true,"heightType":"Exactly"
,"isHeader":false,"borders":{"top":{"color":"#8EADBFF","hasNoneStyle":false
,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"col
or":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"s
hadow":false,"space":0},"right":{"color":"#8EADBFF","hasNoneStyle":false,"l
ineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"colo
r":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"sh
adow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":fals
e,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{
"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shad
ow":false,"space":0},"horizontal":{"color":"#8EADBFF","hasNoneStyle":false,
"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"vertical":{"
color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5

```

```

{"shadow":false,"space":0},"gridBefore":0,"gridBeforeWidth":0,"gridBeforeWidthType":"Point","gridAfter":0,"gridAfterWidth":0,"gridAfterWidthType":"Point"}},{ "cells": [{"blocks": [{"paragraphFormat": {"styleName": "Normal", "listFormat": {}}, "characterFormat": {}, "inlines": [{"characterFormat": {}, "bookmarkType": 0, "name": "GoBack"}, {"characterFormat": {"fontSize": 10, "fontFamily": "Verdana", "fontSizeBidi": 10, "fontFamilyBidi": "Times New Roman"}, "text": "5"}]}]}, "cellFormat": {"borders": {"top": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "left": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "right": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "bottom": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "diagonalDown": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "diagonalUp": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "horizontal": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "vertical": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}}, "shading": {"backgroundColor": "#D9E2F3FF", "foregroundColor": "empty", "textureStyle": "TextureNone"}, "preferredWidth": 13.420000076293945, "preferredWidthType": "Percent", "cellWidth": 64.71214527422465, "columnSpan": 1, "rowSpan": 1, "verticalAlignment": "Top"}, {"blocks": [{"paragraphFormat": {"styleName": "Normal", "listFormat": {}}, "characterFormat": {}, "inlines": [{"characterFormat": {"fontSize": 10, "fontFamily": "Verdana", "fontSizeBidi": 10, "fontFamilyBidi": "Times New Roman"}, "text": "Camembert Pierrot"}]}]}, "cellFormat": {"borders": {"top": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "left": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "right": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "bottom": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "diagonalDown": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "diagonalUp": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "horizontal": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "vertical": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}}, "shading": {"backgroundColor": "#D9E2F3FF", "foregroundColor": "empty", "textureStyle": "TextureNone"}, "preferredWidth": 48.86000061035156, "preferredWidthType": "Percent", "cellWidth": 292.87942351880633, "columnSpan": 1, "rowSpan": 1, "verticalAlignment": "Top"}, {"blocks": [{"paragraphFormat": {"styleName": "Normal", "listFormat": {}}, "characterFormat": {}, "inlines": [{"characterFormat": {"fontSize": 10, "fontFamily": "Verdana", "fontSizeBidi": 10, "fontFamilyBidi": "Times New Roman"}, "text": "46.825"}]}]}, "cellFormat": {"borders": {"top": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "left": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "right": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "bottom": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "diagonalDown": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "diagonalUp": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "horizontal": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "vertical": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}}, "shading": {"backgroundColor": "#D9E2F3FF", "foregroundColor": "empty", "textureStyle": "TextureNone"}, "preferredWidth": 48.86000061035156, "preferredWidthType": "Percent", "cellWidth": 292.87942351880633, "columnSpan": 1, "rowSpan": 1, "verticalAlignment": "Top"}]}]}

```

```

lineWidth":0,"shadow":false,"space":0}},{"shading":{"backgroundColor":"#D9E2F3FF","foregroundColor":"empty","textureStyle":"TextureNone"},"preferredWidth":37.720001220703125,"preferredWidthType":"Percent","cellWidth":117.95841899993776,"columnSpan":1,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":2}},{"rowFormat":{"height":14.399999618530273,"allowBreakAcrossPages":true,"heightType":"Exactly","isHeader":false,"borders":{"top":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"vertical":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0}},"gridBefore":0,"gridBeforeWidth":0,"gridBeforeWidthType":"Point","gridAfter":0,"gridAfterWidth":0,"gridAfterWidthType":"Point"}},{cells":[{"blocks":[{"paragraphFormat":{"styleName":"Normal","listFormat":{},"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"6"}]}],"cellFormat":{"borders":{"top":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0}},"shading":{"backgroundColor":"#FFFFFF","foregroundColor":"empty","textureStyle":"TextureNone"},"preferredWidth":13.420000076293945,"preferredWidthType":"Percent","cellWidth":64.71214527422465,"columnSpan":1,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":0},{blocks":[{"paragraphFormat":{"styleName":"Normal","listFormat":{},"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"Gnocchi di nonna Alice"}]}],"cellFormat":{"borders":{"top":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0}},"shading":{"backgroundColor":"#FFFFFF","foregroundColor":"empty","textureStyle":"TextureNone"},"preferredWidth":48.860000610351

```

```

56,"preferredWidthType":"Percent","cellWidth":292.87942351880633,"columnSpan":1,"rowSpan":1,"verticalAlignment":"Top"},{"columnIndex":1},{
"blocks":[{"paragraphFormat":{"styleName":"Normal","listFormat":{},"characterFormat":{},"inlines":[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"42.593"}]}]},{"cellFormat":{"borders":{"top":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"shading":{"backgroundColor":"#FFFFFF","foregroundColor":"empty","textureStyle":"TextureNone"},"preferredWidth":37.720001220703125,"preferredWidthType":"Percent","cellWidth":117.95841899993776,"columnSpan":1,"rowSpan":1,"verticalAlignment":"Top"},{"columnIndex":2}]},"rowFormat":{"height":14.399999618530273,"allowBreakAcrossPages":true,"heightType":"Exactly","isHeader":false,"borders":{"top":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"vertical":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"gridBefore":0,"gridBeforeWidth":0,"gridBeforeWidthType":"Point","gridAfter":0,"gridAfterWidth":0,"gridAfterWidthType":"Point"}},{
"cells":[{"blocks":[{"paragraphFormat":{"styleName":"Normal","listFormat":{},"characterFormat":{},"inlines":[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"7"}]}]},{"cellFormat":{"borders":{"top":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"shading":{"backgroundColor":"#D9E2F3FF","foregroundColor":"empty","textureStyle":"TextureNone"},"preferredWidth":13.420000076293945,"preferredWidthType":"Percent","cellWidth":64.71214527422465,"columnSpan":1,"rowSpan":1,"verticalAlignment":"Top"},{"blocks":[{"paragraphFormat":{"styleName":"Normal","listFormat":{},"characterF

```

```

format":{},"inlines":[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana",
"fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"Manjimup
Dried
Apples"}]]], "cellFormat":{"borders":{"top":{"color":"#8EAADBFF","hasNoneStyl
e":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"lef
t":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth
":0.5,"shadow":false,"space":0},"right":{"color":"#8EAADBFF","hasNoneStyle":
false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom
":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth"
:0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyl
e":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagon
alUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth"
:0,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":
false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":
{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"sh
adow":false,"space":0},"shading":{"backgroundColor":"#D9E2F3FF","foregroundColor":"empty","textureStyle":"TextureNone"},"preferredWidth":48.86000061035
156,"preferredWidthType":"Percent","cellWidth":292.87942351880633,"columnSpa
n":1,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":1},{ "blocks":[{"pa
ragraphFormat":{"styleName":"Normal","listFormat":{},"characterFormat":{},"
inlines":[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSize
Bidi":10,"fontFamilyBidi":"Times New
Roman"},"text":"41.819"}]]], "cellFormat":{"borders":{"top":{"color":"#8EAADB
FF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false
,"space":0},"left":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Si
ngle","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EAADBFF"
,"hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"s
pace":0},"bottom":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Sin
gle","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000
000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"s
pace":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"N
one","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#000000"
,"hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space
":0},"vertical":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","
lineWidth":0,"shadow":false,"space":0},"shading":{"backgroundColor":"#D9E2F
3FF","foregroundColor":"empty","textureStyle":"TextureNone"},"preferredWidth
":37.720001220703125,"preferredWidthType":"Percent","cellWidth":117.95841899
993776,"columnSpan":1,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":2
}]],"rowFormat":{"height":14.399999618530273,"allowBreakAcrossPages":true,"he
ightType":"Exactly","isHeader":false,"borders":{"top":{"color":"#8EAADBFF","
hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"spa
ce":0},"left":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single"
,"lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EAADBFF","has
NoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space"
:0},"bottom":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single",
"lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000",
"hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space"
:0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None",
"lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#8EAADBFF","h
asNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"spac
e":0},"vertical":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Sing
le","lineWidth":0.5,"shadow":false,"space":0},"gridBefore":0,"gridBeforeWid
th":0,"gridBeforeWidthType":"Point","gridAfter":0,"gridAfterWidth":0,"gridAf
terWidthType":"Point"}]}, {"cells":[{"blocks":[{"paragraphFormat":{"styleName"
:"Normal","listFormat":{},"characterFormat":{},"inlines":[{"characterFormat
":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"
Times New

```

```

Roman"},"text":"8"}]]],"cellFormat":{"borders":{"top":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"shading":{"backgroundColor":"#FFFFFF","foregroundColor":"empty","textureStyle":"TextureNone"},"preferredWidth":13.420000076293945,"preferredWidthType":"Percent","cellWidth":64.71214527422465,"columnSpan":1,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":0},{
"blocks":[{"paragraphFormat":{"styleName":"Normal","listFormat":{},"characterFormat":{},"inlines":[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"Alice Mutton"}]}]}],"cellFormat":{"borders":{"top":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"shading":{"backgroundColor":"#FFFFFF","foregroundColor":"empty","textureStyle":"TextureNone"},"preferredWidth":48.86000061035156,"preferredWidthType":"Percent","cellWidth":292.87942351880633,"columnSpan":1,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":1},{
"blocks":[{"paragraphFormat":{"styleName":"Normal","listFormat":{},"characterFormat":{},"inlines":[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"32.698"}]}]}],"cellFormat":{"borders":{"top":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"shading":{"backgroundColor":"#FFFFFF","foregroundColor":"empty","textureStyle":"TextureNone"},"preferredWidth":37.720001220703125,"preferredWidthType":"Percent","cellWidth":117.95841899993776,"columnSpan":1,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":2}]],"rowFormat":{"height":14.3999999618530273,"allowBreakAcrossPages":true,"heightType":"Exactly","isHeader":false,"borders":{"top":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"spa

```



```

ce":0}, "left": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single",
"lineWidth": 0.5, "shadow": false, "space": 0}, "right": {"color": "#8EADBFF", "has
NoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space"
:0}, "bottom": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single",
"lineWidth": 0.5, "shadow": false, "space": 0}, "diagonalDown": {"color": "#000000",
"hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space"
:0}, "diagonalUp": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None",
"lineWidth": 0, "shadow": false, "space": 0}, "horizontal": {"color": "#8EADBFF", "h
asNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "spac
e": 0}, "vertical": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Sing
le", "lineWidth": 0.5, "shadow": false, "space": 0}}, "gridBefore": 0, "gridBeforeWid
th": 0, "gridBeforeWidthType": "Point", "gridAfter": 0, "gridAfterWidth": 0, "gridAf
terWidthType": "Point"}, {"cells": [{"blocks": [{"paragraphFormat": {"styleName":
"Normal", "listFormat": {}, "characterFormat": {}, "inlines": [{"characterFormat
": {"fontSize": 10, "fontFamily": "Verdana", "fontSizeBidi": 10, "fontFamilyBidi": "
Times New
Roman"}, "text": "9"}]}]}], "cellFormat": {"borders": {"top": {"color": "#8EADBFF", "
hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "spa
ce": 0}, "left": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single"
, "lineWidth": 0.5, "shadow": false, "space": 0}, "right": {"color": "#8EADBFF", "has
NoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space"
:0}, "bottom": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single",
"lineWidth": 0.5, "shadow": false, "space": 0}, "diagonalDown": {"color": "#000000",
"hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space"
:0}, "diagonalUp": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None",
"lineWidth": 0, "shadow": false, "space": 0}, "horizontal": {"color": "#000000", "has
NoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0},
"vertical": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineW
idth": 0, "shadow": false, "space": 0}}, "shading": {"backgroundColor": "#D9E2F3FF",
"foregroundColor": "empty", "textureStyle": "TextureNone", "preferredWidth": 13.
42000076293945, "preferredWidthType": "Percent", "cellWidth": 64.71214527422465
, "columnSpan": 1, "rowSpan": 1, "verticalAlignment": "Top", "columnIndex": 0}, {""bl
ocks": [{"paragraphFormat": {"styleName": "Normal", "listFormat": {}}, "characterF
ormat": {}, "inlines": [{"characterFormat": {"fontSize": 10, "fontFamily": "Verdana
", "fontSizeBidi": 10, "fontFamilyBidi": "Times New Roman"}, "text": "Carnarvon
Tigers"}]}]}], "cellFormat": {"borders": {"top": {"color": "#8EADBFF", "hasNoneStyl
e": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "lef
t": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth"
: 0.5, "shadow": false, "space": 0}, "right": {"color": "#8EADBFF", "hasNoneStyle":
false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "bottom
": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth"
: 0.5, "shadow": false, "space": 0}, "diagonalDown": {"color": "#000000", "hasNoneSty
le": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "diagon
alUp": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth"
: 0, "shadow": false, "space": 0}, "horizontal": {"color": "#000000", "hasNoneStyle":
false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "vertical":
{"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "sh
adow": false, "space": 0}}, "shading": {"backgroundColor": "#D9E2F3FF", "foreground
Color": "empty", "textureStyle": "TextureNone", "preferredWidth": 48.86000061035
156, "preferredWidthType": "Percent", "cellWidth": 292.87942351880633, "columnSpa
n": 1, "rowSpan": 1, "verticalAlignment": "Top", "columnIndex": 1}, {""blocks": [{"pa
ragraphFormat": {"styleName": "Normal", "listFormat": {}}, "characterFormat": {}, "
inlines": [{"characterFormat": {"fontSize": 10, "fontFamily": "Verdana", "fontSize
Bidi": 10, "fontFamilyBidi": "Times New
Roman"}, "text": "29.171"}]}]}], "cellFormat": {"borders": {"top": {"color": "#8EADB
FF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false
, "space": 0}, "left": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Si

```

```

ngle", "lineWidth": 0.5, "shadow": false, "space": 0}, "right": {"color": "#8EADBFF",
"hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "s
pace": 0}, "bottom": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Sin
gle", "lineWidth": 0.5, "shadow": false, "space": 0}, "diagonalDown": {"color": "#000
000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "s
pace": 0}, "diagonalUp": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "N
one", "lineWidth": 0, "shadow": false, "space": 0}, "horizontal": {"color": "#000000"
, "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space
": 0}, "vertical": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "
lineWidth": 0, "shadow": false, "space": 0}}, "shading": {"backgroundColor": "#D9E2F
3FF", "foregroundColor": "empty", "textureStyle": "TextureNone"}, "preferredWidth
": 37.720001220703125, "preferredWidthType": "Percent", "cellWidth": 117.95841899
993776, "columnSpan": 1, "rowSpan": 1, "verticalAlignment": "Top", "columnIndex": 2
}}, "rowFormat": {"height": 14.399999618530273, "allowBreakAcrossPages": true, "he
ightType": "Exactly", "isHeader": false, "borders": {"top": {"color": "#8EADBFF", "
hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "spa
ce": 0}, "left": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single"
, "lineWidth": 0.5, "shadow": false, "space": 0}, "right": {"color": "#8EADBFF", "ha
sNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space
": 0}, "bottom": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single",
"lineWidth": 0.5, "shadow": false, "space": 0}, "diagonalDown": {"color": "#000000",
"hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space
": 0}, "diagonalUp": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None",
"lineWidth": 0, "shadow": false, "space": 0}, "horizontal": {"color": "#8EADBFF", "h
asNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "spac
e": 0}, "vertical": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Sing
le", "lineWidth": 0.5, "shadow": false, "space": 0}}, "gridBefore": 0, "gridBeforeWid
th": 0, "gridBeforeWidthType": "Point", "gridAfter": 0, "gridAfterWidth": 0, "gridAf
terWidthType": "Point"}}, {"cells": [{"blocks": [{"paragraphFormat": {"styleName"
: "Normal", "listFormat": {}, "characterFormat": {}, "inlines": [{"characterFormat
": {"fontSize": 10, "fontFamily": "Verdana", "fontSizeBidi": 10, "fontFamilyBidi": "
Times New
Roman"}, "text": "10"}]}]}, "cellFormat": {"borders": {"top": {"color": "#8EADBFF",
"hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "sp
ace": 0}, "left": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single
", "lineWidth": 0.5, "shadow": false, "space": 0}, "right": {"color": "#8EADBFF", "ha
sNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space
": 0}, "bottom": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single"
, "lineWidth": 0.5, "shadow": false, "space": 0}, "diagonalDown": {"color": "#000000",
"hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space
": 0}, "diagonalUp": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None",
"lineWidth": 0, "shadow": false, "space": 0}, "horizontal": {"color": "#000000", "ha
sNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}
, "vertical": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "line
Width": 0, "shadow": false, "space": 0}}, "shading": {"backgroundColor": "#FFFFFFF
F", "foregroundColor": "empty", "textureStyle": "TextureNone"}, "preferredWidth": 13
.420000076293945, "preferredWidthType": "Percent", "cellWidth": 64.7121452742246
5, "columnSpan": 1, "rowSpan": 1, "verticalAlignment": "Top", "columnIndex": 0}, {"b
locks": [{"paragraphFormat": {"styleName": "Normal", "listFormat": {}, "character
Format": {}, "inlines": [{"characterFormat": {"fontSize": 10, "fontFamily": "Verdan
a", "fontSizeBidi": 10, "fontFamilyBidi": "Times New Roman"}, "text": "Rössle
Sauerkraut."}]}]}, "cellFormat": {"borders": {"top": {"color": "#8EADBFF", "hasNon
eStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}
, "left": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "line
Width": 0.5, "shadow": false, "space": 0}, "right": {"color": "#8EADBFF", "hasNoneSt
yle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "b
ottom": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineW

```



```

idth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0}},"shading":{"backgroundColor":"#FFFFFF","foregroundColor":"empty","textureStyle":"TextureNone"},"preferredWidth":48.86000061035156,"preferredWidthType":"Percent","cellWidth":292.87942351880633,"columnSpan":1,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":1},{ "blocks": [{"paragraphFormat":{"styleName":"Normal","listFormat":{},"characterFormat":{},"inlines":[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"25.696"}]}]},"cellFormat":{"borders":{"top":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0}},"shading":{"backgroundColor":"#FFFFFF","foregroundColor":"empty","textureStyle":"TextureNone"},"preferredWidth":37.720001220703125,"preferredWidthType":"Percent","cellWidth":117.95841899993776,"columnSpan":1,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":2}], "rowFormat":{"height":14.399999618530273,"allowBreakAcrossPages":true,"rightType":"Exactly","isHeader":false,"borders":{"top":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"vertical":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0}},"gridBefore":0,"gridBeforeWidth":0,"gridBeforeWidthType":"Point","gridAfter":0,"gridAfterWidth":0,"gridAfterWidthType":"Point"}]},"grid":[64.71214527422465,292.87942351880633,117.95841899993776],"tableFormat":{"borders":{"top":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"vertical":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","line

```

```

eWidth":0.5,"shadow":false,"space":0}},{"shading":{"backgroundColor":"#FFFFFF",
"foregroundColor":"empty","textureStyle":"TextureNone"},"cellSpacing":0,
"leftIndent":0,"tableAlignment":"Left","topMargin":0,"rightMargin":0.5,"left
Margin":0.5,"bottomMargin":0,"preferredWidth":475.54998779296875,"preferredW
idthType":"Point","bidi":false,"allowAutoFit":true},"description":null,"titl
e":null},{ "paragraphFormat":{"afterSpacing":0,"styleName":"Normal","listForm
at":{}}, {"characterFormat":{"fontFamily":"Calibri","fontColor":"#000000FF","f
ontFamilyBidi":"Calibri"},"inlines":[]}, {"paragraphFormat":{"afterSpacing":0
,"styleName":"Normal","listFormat":{}}, {"characterFormat":{},"inlines":[{"cha
racterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontF
amilyBidi":"Times New Roman"},"text":"The best-selling product of the
company is Cote de Blaye, being part of the Beverages
"}, {"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":1
0,"fontFamilyBidi":"Times New Roman"},"text":"category. The contribution of
this product to the sum of our sales is $
141.396."}]}, {"paragraphFormat":{"afterSpacing":0,"lineSpacing":1,"lineSpaci
ngType":"Multiple","styleName":"Normal","listFormat":{}}, {"characterFormat":{"
fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Tim
es New
Roman"},"inlines":[]}, {"paragraphFormat":{"afterSpacing":0,"lineSpacing":1,"
lineSpacingType":"Multiple","styleName":"Normal","listFormat":{}}, {"character
Format":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyB
idi":"Times New
Roman"},"inlines":[]}, {"paragraphFormat":{"styleName":"Normal","listFormat":
{}}, {"characterFormat":{},"inlines":[{"characterFormat":{},"chartLegend":{"po
sition":"Right","chartTitleArea":{"fontName":"+mn-
lt","fontSize":9,"layout":{"layoutX":0,"layoutY":0},"dataFormat":{"fill":{"f
oreColor":"000000","rgb":"#000000"},"line":{"color":"808080","rgb":"#808080"
}}}}, {"chartTitleArea":{"fontName":"+mn-
lt","fontSize":14,"layout":{"layoutX":0,"layoutY":0},"dataFormat":{"fill":{"
foreColor":"000000","rgb":"#000000"},"line":{"color":"000000","rgb":"#000000
"}}}}, {"chartArea":{"foreColor":"#FFFFFFF","plotArea":{"foreColor":"#000000F
F"},"chartCategory":[{"chartData":[{"yValue":141.396},"categoryXName":"Côte
de Blaye"}, {"chartData":[{"yValue":80.368},"categoryXName":"Thüringer
Rostbratwurst"}, {"chartData":[{"yValue":71.155},"categoryXName":"Raclette
Courdavault"}, {"chartData":[{"yValue":47.234},"categoryXName":"Tarte au
sucre"}, {"chartData":[{"yValue":46.825},"categoryXName":"Camembert
Pierrot"}, {"chartData":[{"yValue":42.593},"categoryXName":"Gnocchi di nonna
Alice"}, {"chartData":[{"yValue":41.819},"categoryXName":"Manjimup Dried
Apples"}, {"chartData":[{"yValue":32.698},"categoryXName":"Alice
Mutton"}, {"chartData":[{"yValue":29.171},"categoryXName":"Carnarvon
Tigers"}, {"chartData":[{"yValue":25.696},"categoryXName":"Rössle
Sauerkraut"}]}, {"chartSeries":[{"dataPoints":[{"fill":{"foreColor":"4472c4","r
gb":"#4472c4"},"line":{"color":"ffffff","rgb":"#ffffff"}}, {"fill":{"foreColo
r":"ed7d31","rgb":"#ed7d31"},"line":{"color":"ffffff","rgb":"#ffffff"}}, {"fi
ll":{"foreColor":"a5a5a5","rgb":"#a5a5a5"},"line":{"color":"ffffff","rgb":"#
ffffff"}}, {"fill":{"foreColor":"ffc000","rgb":"#ffc000"},"line":{"color":"ff
ffff","rgb":"#ffffff"}}, {"fill":{"foreColor":"5b9bd5","rgb":"#5b9bd5"},"line
":{"color":"ffffff","rgb":"#ffffff"}}, {"fill":{"foreColor":"70ad47","rgb":"#
70ad47"},"line":{"color":"ffffff","rgb":"#ffffff"}}, {"fill":{"foreColor":"26
4379","rgb":"#264379"},"line":{"color":"ffffff","rgb":"#ffffff"}}, {"fill":{"
foreColor":"9f480e","rgb":"#9f480e"},"line":{"color":"ffffff","rgb":"#ffffff
"}}, {"fill":{"foreColor":"636363","rgb":"#636363"},"line":{"color":"ffffff",
"rgb":"#ffffff"}}, {"fill":{"foreColor":"9a7200","rgb":"#9a7200"},"line":{"co
lor":"ffffff","rgb":"#ffffff"}}, {"seriesName":"Sales"}]}, {"chartPrimaryCategor
yAxis":{"chartTitle":null,"chartTitleArea":{"layout":{},"dataFormat":{"fill"
:{},"line":{}}},"categoryType":"Automatic","fontSize":11,"fontName":"Calibri

```

```

", "numberFormat": "General", "maximumValue": 0, "minimumValue": 0, "majorUnit": 0, "
hasMajorGridLines": false, "hasMinorGridLines": false, "majorTickMark": "TickMark
_Outside", "minorTickMark": "TickMark_None", "tickLabelPosition": "TickLabelPosi
tion_NextToAxis"}, {"chartPrimaryValueAxis": {"chartTitle": null, "chartTitleArea
": {"layout": {}, "dataFormat": {"fill": {}, "line": {}}, "fontSize": 11, "fontName":
"Calibri", "maximumValue": 0, "minimumValue": 0, "majorUnit": 0, "hasMajorGridLines
": false, "hasMinorGridLines": false, "majorTickMark": "TickMark_Outside", "minorT
ickMark": "TickMark_None", "tickLabelPosition": "TickLabelPosition_NextToAxis"}
, "chartTitle": "Best Selling
Products", "chartType": "Pie", "gapWidth": 0, "overlap": 0, "height": 225, "width": 43
2}}], {"paragraphFormat": {"styleName": "Normal", "listFormat": {}}, {"characterFor
mat": {}, {"inlines": []}, {"paragraphFormat": {"afterSpacing": 0, "lineSpacing": 1, "
lineSpacingType": "Multiple", "styleName": "Normal", "listFormat": {}}, {"character
Format": {"fontSize": 10, "fontFamily": "Verdana", "fontSizeBidi": 10, "fontFamilyB
idi": "Verdana"}, {"inlines": [{"characterFormat": {"fontSize": 10, "fontFamily": "V
erdana", "styleName": "a", "fontSizeBidi": 10, "fontFamilyBidi": "Verdana"}, {"text"
: "According to the above chart, the total count of the selling products is
24 and the average
"}, {"characterFormat": {"fontSize": 10, "fontFamily": "Verdana", "styleName": "a",
"fontSizeBidi": 10, "fontFamilyBidi": "Verdana"}, {"text": "sales attributed to
this product is $ 5.891 with highest sale $ 15.810 in the month of May in
"}, {"characterFormat": {"fontSize": 10, "fontFamily": "Verdana", "styleName": "a",
"fontSizeBidi": 10, "fontFamilyBidi": "Verdana"}, {"text": "2014. In the same
year, in the month of March the same product reached the amount of $
"}, {"characterFormat": {"fontSize": 10, "fontFamily": "Verdana", "styleName": "a",
"fontSizeBidi": 10, "fontFamilyBidi": "Verdana"}, {"text": "15.019. These were the
highest sales of the product among the other products for the year
"}, {"characterFormat": {"fontSize": 10, "fontFamily": "Verdana", "styleName": "a",
"fontSizeBidi": 10, "fontFamilyBidi": "Verdana"}, {"text": "2014."}]]], {"headersFoo
ters": {}}, {"characterFormat": {"bold": false, "italic": false, "fontSize": 11, "fon
tFamily": "Calibri", "underline": "None", "strikethrough": "None", "baselineAlignm
ent": "Normal", "highlightColor": "NoColor", "fontColor": "#000000", "fontSizeBidi
": 11, "fontFamilyBidi": "Calibri"}, {"paragraphFormat": {"leftIndent": 0, "rightInd
ent": 0, "firstLineIndent": 0, "textAlignment": "Left", "beforeSpacing": 0, "afterSp
acing": 8, "lineSpacing": 1.0791666507720947, "lineSpacingType": "Multiple", "list
Format": {}, "bidi": false}, {"defaultTabWidth": 36, "enforcement": false, "hashValue
": "", "saltValue": "", "formatting": false, "protectionType": "NoProtection", "styl
es": [{"name": "Normal", "type": "Paragraph", "paragraphFormat": {"listFormat": {}
}, {"characterFormat": {}, {"next": "Normal"}, {"name": "Heading
1", "type": "Paragraph", "paragraphFormat": {"beforeSpacing": 12, "afterSpacing": 3
, "lineSpacing": 1, "lineSpacingType": "Multiple", "outlineLevel": "Level1", "listF
ormat": {}}, {"characterFormat": {"bold": true, "fontSize": 16, "fontFamily": "Arial"
, "boldBidi": true, "fontSizeBidi": 16, "fontFamilyBidi": "Arial"}, {"basedOn": "Norm
al", "link": "Heading 1 Char", "next": "Normal"}, {"name": "Heading 1
Char", "type": "Character", "characterFormat": {"bold": true, "fontSize": 16, "fontF
amily": "Arial", "boldBidi": true, "fontSizeBidi": 16, "fontFamilyBidi": "Arial"}, {"
basedOn": "Default Paragraph Font"}, {"name": "Default Paragraph
Font", "type": "Character", "characterFormat": {}}, {"name": "Balloon
Text", "type": "Paragraph", "paragraphFormat": {"afterSpacing": 0, "lineSpacing": 1
, "lineSpacingType": "Multiple", "listFormat": {}}, {"characterFormat": {"fontSize"
: 9, "fontFamily": "Segoe UI", "fontSizeBidi": 9, "fontFamilyBidi": "Segoe
UI"}, {"basedOn": "Normal", "link": "Balloon Text Char"}, {"name": "Balloon Text
Char", "type": "Character", "characterFormat": {"fontSize": 9, "fontFamily": "Segoe
UI", "fontSizeBidi": 9, "fontFamilyBidi": "Segoe UI"}, {"basedOn": "Default
Paragraph
Font"}, {"name": "a", "type": "Character", "characterFormat": {}, {"basedOn": "Defaul
t Paragraph Font"}, {"name": "Heading

```

```

2", "type": "Paragraph", "paragraphFormat": { "leftIndent": 0, "rightIndent": 0, "firstLineIndent": 0, "textAlignment": "Left", "beforeSpacing": 2, "afterSpacing": 0, "lineSpacing": 1.0791666507720947, "lineSpacingType": "Multiple", "outlineLevel": "Level2", "listFormat": {} }, "characterFormat": { "fontSize": 13, "fontFamily": "Calibri Light", "fontColor": "#2F5496", "basedOn": "Normal", "link": "Heading 2 Char", "next": "Normal" }, { "name": "Heading 2 Char", "type": "Character", "characterFormat": { "fontSize": 13, "fontFamily": "Calibri Light", "fontColor": "#2F5496", "basedOn": "Default Paragraph Font" }, { "name": "Heading 3", "type": "Paragraph", "paragraphFormat": { "leftIndent": 0, "rightIndent": 0, "firstLineIndent": 0, "textAlignment": "Left", "beforeSpacing": 2, "afterSpacing": 0, "lineSpacing": 1.0791666507720947, "lineSpacingType": "Multiple", "outlineLevel": "Level3", "listFormat": {} }, "characterFormat": { "fontSize": 12, "fontFamily": "Calibri Light", "fontColor": "#1F3763", "basedOn": "Normal", "link": "Heading 3 Char", "next": "Normal" }, { "name": "Heading 3 Char", "type": "Character", "characterFormat": { "fontSize": 12, "fontFamily": "Calibri Light", "fontColor": "#1F3763", "basedOn": "Default Paragraph Font" }, { "name": "Heading 4", "type": "Paragraph", "paragraphFormat": { "leftIndent": 0, "rightIndent": 0, "firstLineIndent": 0, "textAlignment": "Left", "beforeSpacing": 2, "afterSpacing": 0, "lineSpacing": 1.0791666507720947, "lineSpacingType": "Multiple", "outlineLevel": "Level4", "listFormat": {} }, "characterFormat": { "italic": true, "fontFamily": "Calibri Light", "fontColor": "#2F5496", "basedOn": "Normal", "link": "Heading 4 Char", "next": "Normal" }, { "name": "Heading 4 Char", "type": "Character", "characterFormat": { "italic": true, "fontFamily": "Calibri Light", "fontColor": "#2F5496", "basedOn": "Default Paragraph Font" }, { "name": "Heading 5", "type": "Paragraph", "paragraphFormat": { "leftIndent": 0, "rightIndent": 0, "firstLineIndent": 0, "textAlignment": "Left", "beforeSpacing": 2, "afterSpacing": 0, "lineSpacing": 1.0791666507720947, "lineSpacingType": "Multiple", "outlineLevel": "Level5", "listFormat": {} }, "characterFormat": { "fontFamily": "Calibri Light", "fontColor": "#2F5496", "basedOn": "Normal", "link": "Heading 5 Char", "next": "Normal" }, { "name": "Heading 5 Char", "type": "Character", "characterFormat": { "fontFamily": "Calibri Light", "fontColor": "#2F5496", "basedOn": "Default Paragraph Font" }, { "name": "Heading 6", "type": "Paragraph", "paragraphFormat": { "leftIndent": 0, "rightIndent": 0, "firstLineIndent": 0, "textAlignment": "Left", "beforeSpacing": 2, "afterSpacing": 0, "lineSpacing": 1.0791666507720947, "lineSpacingType": "Multiple", "outlineLevel": "Level6", "listFormat": {} }, "characterFormat": { "fontFamily": "Calibri Light", "fontColor": "#1F3763", "basedOn": "Normal", "link": "Heading 6 Char", "next": "Normal" }, { "name": "Heading 6 Char", "type": "Character", "characterFormat": { "fontFamily": "Calibri Light", "fontColor": "#1F3763", "basedOn": "Default Paragraph Font" } } ], "lists": [], "abstractLists": [] };
documenteditor.open(JSON.stringify(sfdt));

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <!-- EJ2 Document Editor dependent theme -->

```

```

<link href="http://cdn.syncfusion.com/ej2/ej2-base/styles/material.css"
rel="stylesheet" type="text/css">
<link href="http://cdn.syncfusion.com/ej2/ej2-
buttons/styles/material.css" rel="stylesheet" type="text/css">
<link href="http://cdn.syncfusion.com/ej2/ej2-
inputs/styles/material.css" rel="stylesheet" type="text/css">
<link href="http://cdn.syncfusion.com/ej2/ej2-
popups/styles/material.css" rel="stylesheet" type="text/css">
<link href="http://cdn.syncfusion.com/ej2/ej2-lists/styles/material.css"
rel="stylesheet" type="text/css">
<link href="http://cdn.syncfusion.com/ej2/ej2-
navigations/styles/material.css" rel="stylesheet" type="text/css">
<link href="http://cdn.syncfusion.com/ej2/ej2-
splitbuttons/styles/material.css" rel="stylesheet" type="text/css">
<link href="http://cdn.syncfusion.com/ej2/ej2-
dropdowns/styles/material.css" rel="stylesheet" type="text/css">
<!-- EJ2 Document Editor theme -->
<link href="http://cdn.syncfusion.com/ej2/ej2-
documenteditor/styles/material.css" rel="stylesheet" type="text/css">
<!-- EJ2 Document Editor Javascripts dependent -->
<script src="http://cdn.syncfusion.com/ej2/ej2-base/dist/global/ej2-
base.min.js" type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/ej2/ej2-file-
utils/dist/global/ej2-file-utils.min.js" type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/ej2/ej2-
compression/dist/global/ej2-compression.min.js"
type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/ej2/ej2-buttons/dist/global/ej2-
buttons.min.js" type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/ej2/ej2-popups/dist/global/ej2-
popups.min.js" type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/ej2/ej2-
splitbuttons/dist/global/ej2-splitbuttons.min.js"
type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/ej2/ej2-inputs/dist/global/ej2-
inputs.min.js" type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/ej2/ej2-lists/dist/global/ej2-
lists.min.js" type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/ej2/ej2-data/dist/global/ej2-
data.min.js" type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/ej2/ej2-
navigations/dist/global/ej2-navigations.min.js"
type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/ej2/ej2-
dropdowns/dist/global/ej2-dropdowns.min.js" type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/ej2/ej2-
documenteditor/dist/global/ej2-documenteditor.min.js"
type="text/javascript"></script>
<!-- TypeScripts dependent -->
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
documenteditor/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>

```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <!--Element which will render as DocumentEditor -->
        <div id="DocumentEditor" style="height: 412px"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Supported Chart Types

The following chart types are supported in Document Editor

- Scatter_Markers
- Bubble
- Area
- Area_Stacked
- AreaStacked100
- Bar_Clustered
- Bar_Stacked
- BarStacked100
- Column_Clustered
- Column_Stacked
- ColumnStacked100
- Pie
- Doughnut
- Line
- Line_Markers
- LineMarkersStacked
- LineMarkersStacked_100
- Line_Stacked
- LineStacked100

Content control in ##Platform_Name## Document editor control

Document Editor provides content control preservation support (i.e.) Content control present in the input document is preserved upon saving.

Content controls can be categorized based on its occurrence in a document as follows,

InlineContentControl: Among inline content inside, as a child of a paragraph.

BlockContentControl: Among paragraphs and tables, as a child of a Body, HeaderFooter.

Types of Content Controls

- Rich Text
- Plain Text
- Check Box
- Date picker
- Drop-Down List and Combo Box
- Picture

Note: Content control with custom XML mapping of file type WordML is converted as normal Rich Text Content Control to provide lossless round-tripping upon saving.

Restrict editing in ##Platform_Name## Document editor control

Document Editor provides support to restrict editing. When the protected document includes range permission, then unique user or user group only authorized to edit separate text area.

Set current user

You can use the [currentUser](#) property to authorize the current document user by name, email, or user group name.

The following code shows how to set currentUser

```
`ts
documentEditor.currentUser = 'engineer@mycompany.com';
`
```

Highlighting the text area

You can highlight the editable region of the current user using the [userColor](#) property.

The following code shows how to set userColor.

```
`ts
documentEditor.userColor = '#fff000';
`
```

Restrict Editing Pane

Restrict Editing Pane provides the following options to manage the document:

- To apply formatting restrictions to the current document, select the allow formatting check box.
- To apply editing restrictions to the current document, select the read only check box.
- To add users to the current document, select more users option and add user from the popup dialog.
- To include range permission to the current document, select parts of the document and choose users who are allowed to freely edit them from the listed check box.
- To apply the chosen editing restrictions, click the **YES, START ENFORCING PROTECTION** button. A dialog box displays asking for a password to protect.
- To stop protection, select **STOP PROTECTION** button. A dialog box displays asking for a password to stop protection.

The following code shows Restrict Editing Pane. To unprotect the document, use password '123'.

INDEX.JS

```
var documenteditorContainer = new
ej.documenteditor.DocumentEditorContainer({ enableToolbar: true });

ej.documenteditor.DocumentEditorContainer.Inject(ej.documenteditor.Toolbar);
documenteditorContainer.serviceUrl =
'https://services.syncfusion.com/js/production/api/documenteditor/';
//DocumentEditorContainer control rendering starts
documenteditorContainer.appendTo('#DocumentEditor');
var sfdt
={ "sections": [ { "blocks": [ { "characterFormat": { "fontSize": 14.0, "fontSizeBidi":
14.0 }, "paragraphFormat": { "lineSpacing": 32.0, "lineSpacingType": "Exactly", "sty
leName": "Normal" }, "inlines": [ { "text": "Name", "characterFormat": { "bold": true, "
fontSize": 14.0, "boldBidi": true, "fontSizeBidi": 14.0 }, { "text": ":", "characterF
ormat": { "fontSize": 14.0, "fontSizeBidi": 14.0 } } ] }, { "rows": [ { "rowFormat": { "allo
wBreakAcrossPages": true, "isHeader": false, "height": 20.0, "heightType": "AtLeast
", "borders": { "left": { "lineStyle": "None", "lineWidth": 0.0, "shadow": false, "spac
e": 0.0, "hasNoneStyle": false }, "right": { "lineStyle": "None", "lineWidth": 0.0, "sh
adow": false, "space": 0.0, "hasNoneStyle": false }, "top": { "lineStyle": "None", "lin
eWidth": 0.0, "shadow": false, "space": 0.0, "hasNoneStyle": false }, "bottom": { "line
Style": "None", "lineWidth": 0.0, "shadow": false, "space": 0.0, "hasNoneStyle": fals
e }, "vertical": { "lineStyle": "None", "lineWidth": 0.0, "shadow": false, "space": 0.0
, "hasNoneStyle": false }, "horizontal": { "lineStyle": "None", "lineWidth": 0.0, "sha
dow": false, "space": 0.0, "hasNoneStyle": false }, "diagonalDown": { "lineStyle": "No
ne", "lineWidth": 0.0, "shadow": false, "space": 0.0, "hasNoneStyle": false }, "diagon
alUp": { "lineStyle": "None", "lineWidth": 0.0, "shadow": false, "space": 0.0, "hasNon
eStyle": false } } ] }, { "cells": [ { "blocks": [ { "paragraphFormat": { "styleName": "Normal
"}, "inlines": [ { "editRangeId": "1348272392", "columnFirst": 0, "columnLast": 0, "us
er": "engineer@mycompany.com", { "text": "Enter
name", { "editRangeId": "1348272392", "editableRangeStart": { "editRangeId": "1348
272392", "columnFirst": 0, "columnLast": 0, "user": "engineer@mycompany.com" } } ] } ] },
"cellFormat": { "columnSpan": 1, "rowSpan": 1, "preferredWidth": 467.5, "preferredWi
dthType": "Point", "verticalAlignment": "Center", "isSamePaddingAsTable": true, "b
orders": { "left": { "lineStyle": "None", "lineWidth": 0.0, "shadow": false, "space": 0
.0, "hasNoneStyle": false }, "right": { "lineStyle": "None", "lineWidth": 0.0, "shadow
": false, "space": 0.0, "hasNoneStyle": false }, "top": { "lineStyle": "None", "lineWid
th": 0.0, "shadow": false, "space": 0.0, "hasNoneStyle": false }, "bottom": { "lineStyl
e": "None", "lineWidth": 0.0, "shadow": false, "space": 0.0, "hasNoneStyle": false }, "
vertical": { "lineStyle": "None", "lineWidth": 0.0, "shadow": false, "space": 0.0, "ha
sNoneStyle": false }, "horizontal": { "lineStyle": "None", "lineWidth": 0.0, "shadow
": false, "space": 0.0, "hasNoneStyle": false }, "diagonalDown": { "lineStyle": "None",
"lineWidth": 0.0, "shadow": false, "space": 0.0, "hasNoneStyle": false }, "diagonalUp
": { "lineStyle": "None", "lineWidth": 0.0, "shadow": false, "space": 0.0, "hasNoneStyl
e": false } } ] } ] }, { "title": null, "description": null, "tableFormat": { "allowAutoFi
t": true, "leftIndent": 0.0, "tableAlignment": "Left", "preferredWidthType": "Auto"
, "borders": { "left": { "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "spa
ce": 0.0, "hasNoneStyle": false }, "right": { "lineStyle": "Single", "lineWidth": 0.5,
"shadow": false, "space": 0.0, "hasNoneStyle": false }, "top": { "lineStyle": "Single"
, "lineWidth": 0.5, "shadow": false, "space": 0.0, "hasNoneStyle": false }, "bottom": {
"lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0.0, "hasNoneStyl
e": false }, "vertical": { "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "s
pace": 0.0, "hasNoneStyle": false }, "horizontal": { "lineStyle": "Single", "lineWidt
h": 0.5, "shadow": false, "space": 0.0, "hasNoneStyle": false }, "diagonalDown": { "lin
eStyle": "None", "lineWidth": 0.0, "shadow": false, "space": 0.0, "hasNoneStyle": fal
```



```

se}, "diagonalUp": {"lineStyle": "None", "lineWidth": 0.0, "shadow": false, "space":
0.0, "hasNoneStyle": false}, "bidi": false}, {"characterFormat": {"bold": true, "f
ontSize": 14.0, "boldBidi": true, "fontSizeBidi": 14.0}, "paragraphFormat": {"lineS
pacing": 32.0, "lineSpacingType": "Exactly", "styleName": "Normal"}, "inlines": [{"
text": "Designation:", "characterFormat": {"bold": true, "fontSize": 14.0, "boldBidi
": true, "fontSizeBidi": 14.0}}]}, {"rows": [{"rowFormat": {"allowBreakAcrossPage
s": true, "isHeader": false, "height": 20.0, "heightType": "AtLeast", "borders": {"le
ft": {"lineStyle": "None", "lineWidth": 0.0, "shadow": false, "space": 0.0, "hasNoneS
tyle": false}, "right": {"lineStyle": "None", "lineWidth": 0.0, "shadow": false, "spa
ce": 0.0, "hasNoneStyle": false}, "top": {"lineStyle": "None", "lineWidth": 0.0, "sha
dow": false, "space": 0.0, "hasNoneStyle": false}, "bottom": {"lineStyle": "None", "l
ineWidth": 0.0, "shadow": false, "space": 0.0, "hasNoneStyle": false}, "vertical": {"
lineStyle": "None", "lineWidth": 0.0, "shadow": false, "space": 0.0, "hasNoneStyle":
false}, "horizontal": {"lineStyle": "None", "lineWidth": 0.0, "shadow": false, "spac
e": 0.0, "hasNoneStyle": false}, "diagonalDown": {"lineStyle": "None", "lineWidth":
0.0, "shadow": false, "space": 0.0, "hasNoneStyle": false}, "diagonalUp": {"lineStyl
e": "None", "lineWidth": 0.0, "shadow": false, "space": 0.0, "hasNoneStyle": false}}
}, "cells": [{"blocks": [{"paragraphFormat": {"styleName": "Normal"}, "inlines": [{"
editRangeId": "808933422", "columnFirst": 0, "columnLast": 0, "user": "engineer@myc
ompany.com"}], {"text": "Enter
designation"}, {"editRangeId": "808933422", "editableRangeStart": {"editRangeId":
"808933422", "columnFirst": 0, "columnLast": 0, "user": "engineer@mycompany.com"}
}]}], "cellFormat": {"columnSpan": 1, "rowSpan": 1, "preferredWidth": 467.5, "prefer
redWidthType": "Point", "verticalAlignment": "Center", "isSamePaddingAsTable": true
, "borders": {"left": {"lineStyle": "None", "lineWidth": 0.0, "shadow": false, "spa
ce": 0.0, "hasNoneStyle": false}, "right": {"lineStyle": "None", "lineWidth": 0.0, "s
hadow": false, "space": 0.0, "hasNoneStyle": false}, "top": {"lineStyle": "None", "li
neWidth": 0.0, "shadow": false, "space": 0.0, "hasNoneStyle": false}, "bottom": {"lin
eStyle": "None", "lineWidth": 0.0, "shadow": false, "space": 0.0, "hasNoneStyle": fal
se}, "vertical": {"lineStyle": "None", "lineWidth": 0.0, "shadow": false, "space": 0.
0, "hasNoneStyle": false}, "horizontal": {"lineStyle": "None", "lineWidth": 0.0, "sh
adow": false, "space": 0.0, "hasNoneStyle": false}, "diagonalDown": {"lineStyle": "N
one", "lineWidth": 0.0, "shadow": false, "space": 0.0, "hasNoneStyle": false}, "diago
nalUp": {"lineStyle": "None", "lineWidth": 0.0, "shadow": false, "space": 0.0, "hasNo
neStyle": false}}]}], "title": null, "description": null, "tableFormat": {"allowA
utoFit": true, "leftIndent": 0.0, "tableAlignment": "Left", "preferredWidthType": "
Auto", "borders": {"left": {"lineStyle": "Single", "lineWidth": 0.5, "shadow": false
, "space": 0.0, "hasNoneStyle": false}, "right": {"lineStyle": "Single", "lineWidth"
: 0.5, "shadow": false, "space": 0.0, "hasNoneStyle": false}, "top": {"lineStyle": "Si
ngle", "lineWidth": 0.5, "shadow": false, "space": 0.0, "hasNoneStyle": false}, "bott
om": {"lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0.0, "hasNon
eStyle": false}, "vertical": {"lineStyle": "Single", "lineWidth": 0.5, "shadow": fal
se, "space": 0.0, "hasNoneStyle": false}, "horizontal": {"lineStyle": "Single", "lin
eWidth": 0.5, "shadow": false, "space": 0.0, "hasNoneStyle": false}, "diagonalDown":
{"lineStyle": "None", "lineWidth": 0.0, "shadow": false, "space": 0.0, "hasNoneStyle
": false}, "diagonalUp": {"lineStyle": "None", "lineWidth": 0.0, "shadow": false, "sp
ace": 0.0, "hasNoneStyle": false}, "bidi": false}, {"characterFormat": {"bold": tr
ue, "fontSize": 14.0, "boldBidi": true, "fontSizeBidi": 14.0}, "paragraphFormat": {"
lineSpacing": 32.0, "lineSpacingType": "Exactly", "styleName": "Normal"}, "inlines
": [{"text": "Email
Address:", "characterFormat": {"bold": true, "fontSize": 14.0, "boldBidi": true, "fo
ntSizeBidi": 14.0}}, {"name": "_GoBack", "bookmarkType": 0}, {"name": "_GoBack", "bo
okmarkType": 1}]}], {"rows": [{"rowFormat": {"allowBreakAcrossPages": true, "isHead
er": false, "height": 20.0, "heightType": "AtLeast", "borders": {"left": {"lineStyle
": "None", "lineWidth": 0.0, "shadow": false, "space": 0.0, "hasNoneStyle": false}, "r
ight": {"lineStyle": "None", "lineWidth": 0.0, "shadow": false, "space": 0.0, "hasNon
eStyle": false}, "top": {"lineStyle": "None", "lineWidth": 0.0, "shadow": false, "spa

```

```

ce":0.0,"hasNoneStyle":false},"bottom":{"lineStyle":"None","lineWidth":0.0,"
shadow":false,"space":0.0,"hasNoneStyle":false},"vertical":{"lineStyle":"Non
e","lineWidth":0.0,"shadow":false,"space":0.0,"hasNoneStyle":false},"horizon
tal":{"lineStyle":"None","lineWidth":0.0,"shadow":false,"space":0.0,"hasNone
Style":false},"diagonalDown":{"lineStyle":"None","lineWidth":0.0,"shadow":fa
lse,"space":0.0,"hasNoneStyle":false},"diagonalUp":{"lineStyle":"None","line
Width":0.0,"shadow":false,"space":0.0,"hasNoneStyle":false}}},"cells":[{"blo
cks":[{"paragraphFormat":{"styleName":"Normal"},"inlines":[{"editRangeId":"8
10441411","columnFirst":0,"columnLast":0,"user":"engineer@mycompany.com"}},{
"text":"Enter email
address"}},{editRangeId":"810441411","editableRangeStart":{"editRangeId":"81
0441411","columnFirst":0,"columnLast":0,"user":"engineer@mycompany.com"}]}]
,"cellFormat":{"columnSpan":1,"rowSpan":1,"preferredWidth":467.5,"preferredW
idthType":"Point","verticalAlignment":"Center","isSamePaddingAsTable":true,"
borders":{"left":{"lineStyle":"None","lineWidth":0.0,"shadow":false,"space":
0.0,"hasNoneStyle":false},"right":{"lineStyle":"None","lineWidth":0.0,"shado
w":false,"space":0.0,"hasNoneStyle":false},"top":{"lineStyle":"None","lineWi
dth":0.0,"shadow":false,"space":0.0,"hasNoneStyle":false},"bottom":{"lineSty
le":"None","lineWidth":0.0,"shadow":false,"space":0.0,"hasNoneStyle":false},
"vertical":{"lineStyle":"None","lineWidth":0.0,"shadow":false,"space":0.0,"h
asNoneStyle":false},"horizontal":{"lineStyle":"None","lineWidth":0.0,"shadow
":false,"space":0.0,"hasNoneStyle":false},"diagonalDown":{"lineStyle":"None"
,"lineWidth":0.0,"shadow":false,"space":0.0,"hasNoneStyle":false},"diagonalU
p":{"lineStyle":"None","lineWidth":0.0,"shadow":false,"space":0.0,"hasNoneSt
yle":false}}}}}},"title":null,"description":null,"tableFormat":{"allowAutoF
it":true,"leftIndent":0.0,"tableAlignment":"Left","preferredWidthType":"Auto
","borders":{"left":{"lineStyle":"Single","lineWidth":0.5,"shadow":false,"sp
ace":0.0,"hasNoneStyle":false},"right":{"lineStyle":"Single","lineWidth":0.5
,"shadow":false,"space":0.0,"hasNoneStyle":false},"top":{"lineStyle":"Single
","lineWidth":0.5,"shadow":false,"space":0.0,"hasNoneStyle":false},"bottom":
{"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0.0,"hasNoneSty
le":false},"vertical":{"lineStyle":"Single","lineWidth":0.5,"shadow":false,"
space":0.0,"hasNoneStyle":false},"horizontal":{"lineStyle":"Single","lineWid
th":0.5,"shadow":false,"space":0.0,"hasNoneStyle":false},"diagonalDown":{"li
neStyle":"None","lineWidth":0.0,"shadow":false,"space":0.0,"hasNoneStyle":fa
lse},"diagonalUp":{"lineStyle":"None","lineWidth":0.0,"shadow":false,"space"
:0.0,"hasNoneStyle":false}},"bidi":false}},{"characterFormat":{"bold":true,"
fontSize":14.0,"boldBidi":true,"fontSizeBidi":14.0},"paragraphFormat":{"line
Spacing":32.0,"lineSpacingType":"Exactly","styleName":"Normal"},"inlines":[{"
"text":"Feedbacks:",characterFormat":{"bold":true,"fontSize":14.0,"boldBidi
":true,"fontSizeBidi":14.0}}]}},{rows":[{"rowFormat":{"allowBreakAcrossPages
":true,"isHeader":false,"height":20.0,"heightType":"AtLeast","borders":{"lef
t":{"lineStyle":"None","lineWidth":0.0,"shadow":false,"space":0.0,"hasNoneSt
yle":false},"right":{"lineStyle":"None","lineWidth":0.0,"shadow":false,"spac
e":0.0,"hasNoneStyle":false},"top":{"lineStyle":"None","lineWidth":0.0,"shad
ow":false,"space":0.0,"hasNoneStyle":false},"bottom":{"lineStyle":"None","li
neWidth":0.0,"shadow":false,"space":0.0,"hasNoneStyle":false},"vertical":{"l
ineStyle":"None","lineWidth":0.0,"shadow":false,"space":0.0,"hasNoneStyle":f
alse},"horizontal":{"lineStyle":"None","lineWidth":0.0,"shadow":false,"space
":0.0,"hasNoneStyle":false},"diagonalDown":{"lineStyle":"None","lineWidth":0
.0,"shadow":false,"space":0.0,"hasNoneStyle":false},"diagonalUp":{"lineStyle
":"None","lineWidth":0.0,"shadow":false,"space":0.0,"hasNoneStyle":false}}},
"cells":[{"blocks":[{"paragraphFormat":{"styleName":"Normal"},"inlines":[{"e
ditRangeId":"1016946268","columnFirst":0,"columnLast":0,"user":"manager@myco
mpany.com"}},{text":"Enter the
feedbacks"}},{editRangeId":"1016946268","editableRangeStart":{"editRangeId":
"1016946268","columnFirst":0,"columnLast":0,"user":"manager@mycompany.com"}]}]

```

[illegible]

```

lUp":{"lineStyle":"None","lineWidth":0.0,"shadow":false,"space":0.0,"hasNone
Style":false}}}}]]], "title":null, "description":null, "tableFormat":{"allowAut
oFit":true, "leftIndent":0.0, "tableAlignment":"Left", "preferredWidthType":"Au
to", "borders":{"left":{"lineStyle":"Single", "lineWidth":0.5, "shadow":false, "
space":0.0, "hasNoneStyle":false}, "right":{"lineStyle":"Single", "lineWidth":0
.5, "shadow":false, "space":0.0, "hasNoneStyle":false}, "top":{"lineStyle":"Sing
le", "lineWidth":0.5, "shadow":false, "space":0.0, "hasNoneStyle":false}, "bottom
":{"lineStyle":"Single", "lineWidth":0.5, "shadow":false, "space":0.0, "hasNoneS
tyle":false}, "vertical":{"lineStyle":"Single", "lineWidth":0.5, "shadow":false
, "space":0.0, "hasNoneStyle":false}, "horizontal":{"lineStyle":"Single", "lineW
idth":0.5, "shadow":false, "space":0.0, "hasNoneStyle":false}, "diagonalDown":{"
lineStyle":"None", "lineWidth":0.0, "shadow":false, "space":0.0, "hasNoneStyle":
false}, "diagonalUp":{"lineStyle":"None", "lineWidth":0.0, "shadow":false, "spac
e":0.0, "hasNoneStyle":false}, "bidi":false}}, {"paragraphFormat":{"styleName"
:"Normal"}, "inlines":[]}}, "headersFooters":{"header":{"blocks":[{"paragraphF
ormat":{"styleName":"Header"}, "inlines":[{"text":"Employee's Details
"}]}]}}, "sectionFormat":{"headerDistance":36.0, "footerDistance":36.0, "pageWi
dth":612.0, "pageHeight":792.0, "leftMargin":72.0, "rightMargin":72.0, "topMargi
n":72.0, "bottomMargin":72.0, "differentFirstPage":false, "differentOddAndEvenP
ages":false, "bidi":false}}, "characterFormat":{"fontSize":11.0, "fontFamily":
"Calibri", "fontSizeBidi":11.0, "fontFamilyBidi":"Calibri"}, "paragraphFormat":
{"afterSpacing":8.0, "lineSpacing":1.0791666507720947, "lineSpacingType":"Mult
iple"}, "background":{"color":"#FFFFFFF"}, "styles":[{"type":"Paragraph", "nam
e":"Normal", "next":"Normal"}, {"type":"Character", "name":"Default Paragraph
Font"}, {"type":"Paragraph", "name":"List
Paragraph", "basedOn":"Normal", "paragraphFormat":{"leftIndent":36.0, "contextu
alSpacing":true}}, {"type":"Paragraph", "name":"Header", "basedOn":"Normal", "ne
xt":"Normal", "link":"Header
Char", "paragraphFormat":{"afterSpacing":0.0, "lineSpacing":1.0, "lineSpacingTy
pe":"Multiple", "tabs":[{"tabJustification":"Center", "position":234.0, "tabLea
der":"None", "deletePosition":0.0}, {"tabJustification":"Right", "position":468
.0, "tabLeader":"None", "deletePosition":0.0}]}], {"type":"Character", "name":"H
eader Char", "basedOn":"Default Paragraph
Font"}, {"type":"Paragraph", "name":"Footer", "basedOn":"Normal", "link":"Footer
Char", "paragraphFormat":{"afterSpacing":0.0, "lineSpacing":1.0, "lineSpacingTy
pe":"Multiple", "tabs":[{"tabJustification":"Center", "position":234.0, "tabLea
der":"None", "deletePosition":0.0}, {"tabJustification":"Right", "position":468
.0, "tabLeader":"None", "deletePosition":0.0}]}], {"type":"Character", "name":"F
ooter Char", "basedOn":"Default Paragraph
Font"}], "defaultTabWidth":36.0, "formatting":false, "protectionType":"ReadOnly
", "enforcement":true, "hashValue":"TQGuJuLceQCe234Ygx4q6NFgHpRMfilhjFTojyKzbQ
Okwk+ckEM9CjNIdkiUhSR/e/7sfMxO4sbPcg/DBzztMg==", "saltValue":"FXbkrlRtDIIIZfw
lM71dMg=="};
documenteditorContainer.documentEditor.open(JSON.stringify(sfdt));
documenteditorContainer.documentEditor.currentUser =
'engineer@mycompany.com';

```

INDEX.HTML

```

<!DOCTYPE html><html xmlns="http://www.w3.org/1999/xhtml"><head>
  <title>Essential JS 2</title>
  <!-- EJ2 Document Editor dependent theme -->
  <link href="http://cdn.syncfusion.com/ej2/ej2-
base/styles/material.css" rel="stylesheet" type="text/css">
  <link href="http://cdn.syncfusion.com/ej2/ej2-
buttons/styles/material.css" rel="stylesheet" type="text/css">

```

```

<link href="http://cdn.syncfusion.com/ej2/ej2-
inputs/styles/material.css" rel="stylesheet" type="text/css">
<link href="http://cdn.syncfusion.com/ej2/ej2-
popups/styles/material.css" rel="stylesheet" type="text/css">
<link href="http://cdn.syncfusion.com/ej2/ej2-
lists/styles/material.css" rel="stylesheet" type="text/css">
<link href="http://cdn.syncfusion.com/ej2/ej2-
navigations/styles/material.css" rel="stylesheet" type="text/css">
<link href="http://cdn.syncfusion.com/ej2/ej2-
splitbuttons/styles/material.css" rel="stylesheet" type="text/css">
<link href="http://cdn.syncfusion.com/ej2/ej2-
dropdowns/styles/material.css" rel="stylesheet" type="text/css">
<!-- EJ2 Document Editor theme -->
<link href="http://cdn.syncfusion.com/ej2/ej2-
documenteditor/styles/material.css" rel="stylesheet" type="text/css">
<!-- EJ2 Document Editor Javascript dependent -->
<script src="http://cdn.syncfusion.com/ej2/ej2-
base/dist/global/ej2-base.min.js" type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/ej2/ej2-file-
utils/dist/global/ej2-file-utils.min.js" type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/ej2/ej2-
compression/dist/global/ej2-compression.min.js"
type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/ej2/ej2-
buttons/dist/global/ej2-buttons.min.js" type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/ej2/ej2-
popups/dist/global/ej2-popups.min.js" type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/ej2/ej2-
splitbuttons/dist/global/ej2-splitbuttons.min.js"
type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/ej2/ej2-
inputs/dist/global/ej2-inputs.min.js" type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/ej2/ej2-
lists/dist/global/ej2-lists.min.js" type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/ej2/ej2-
data/dist/global/ej2-data.min.js" type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/ej2/ej2-
navigations/dist/global/ej2-navigations.min.js"
type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/ej2/ej2-
dropdowns/dist/global/ej2-dropdowns.min.js" type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/ej2/ej2-
documenteditor/dist/global/ej2-documenteditor.min.js"
type="text/javascript"></script>
<!-- TypeScripts Dependent -->
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
documenteditor/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/fabric.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
<!--element which is going to render-->
<div id="DocumentEditor" style="height:420px">
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [How to protect the document in form filling mode](#)
- [How to protect the document in comments only mode](#)
- [How to protect the document in track changes only mode](#)

Spell check in ##Platform_Name## Document editor control

Document Editor supports performing spell checking for any input text. You can perform spell checking for the text in Document Editor and it will provide suggestions for the mis-spelled words through dialog and in context menu. Document editor's spell checker is compatible with [hunspell dictionary files](#).

`ts

```
import { DocumentEditorContainer, Toolbar, SpellChecker } from '@syncfusion/ej2-documenteditor';
```

```
DocumentEditorContainer.Inject(Toolbar);
```

```
let container: DocumentEditorContainer = new DocumentEditorContainer({
```

```
    enableToolbar: true, enableSpellCheck: true
```

```
});
```

```
container.appendTo('#container');
```

```
//Accessing spell checker.
```

```
let spellChecker: SpellChecker = container.documentEditor.spellChecker;
```

```
//Set language id to map dictionary in server side.;
spellChecker.languageID = 1033;
spellChecker.removeUnderline = false;
//Allow suggestion for miss spelled word/
spellChecker.allowSpellCheckAndSuggestion = true;
`ts
```

Note: Document Editor requires server-side dependencies for spell check configuration.

Refer to the [Document Editor Web API service projects from GitHub](#) link for configuring spell checker in server-side. To know about server-side dependencies, please refer this [page](#).

Features

- Supports context menu suggestions.
- Provides built-in options to Ignore, Ignore All, Change, Change All for error words in spell checker dialog.

Enable SpellCheck

To enable spell check in DocumentEditor, set [enableSpellCheck](#) property as `true` and then configure SpellCheckSettings.

Disable SpellCheck

To disable spell check in DocumentEditor, set [enableSpellCheck](#) property as `false` or remove [enableSpellCheck](#) property initialization code. The default value of this property is false.

Spell check settings

Remove Underline

By default, mis-spelled words are marked with squiggly line. You can also disable this behavior by enabling the [removeUnderline](#) API and now, the squiggly lines will never be rendered for mis-spelled words.

```
`ts
documentEditor.spellChecker.removeUnderline = false;
`ts
```

AllowSpellCheckAndSuggestion

By default, on performing spell check in Document Editor, both spelling and suggestions of the mis-spelled words will be retrieved, and this mis-spelled words can be corrected through context menu suggestions. You can modify this behavior using the [allowSpellCheckAndSuggestion](#) API, which will perform only spell check.

```
`ts
documentEditor.spellChecker.allowSpellCheckAndSuggestion = false;
`ts
```


LanguageID

Document Editor provides multi-language spell check support. You can add as many languages (dictionaries) in the server-side and to use that language for spell checking in Document Editor, it must be matched with [languageID](#) you pass in the Document Editor.

```
`ts
```

```
documentEditor.spellChecker.languageID = 1033; //LCID of "en-us";
```

```
,
```

EnableOptimizedSpellCheck

Document Editor provides option to spellcheck page by page when loading the documents. The default value of this property is false, so when opening the document spellcheck web API will be called for each word in the document. To optimize the frequency of spellcheck web API calls, you can enable this property.

The following code example illustrates how to enable optimized spell checking.

```
`ts
```

```
documentEditor.spellChecker.enableOptimizedSpellCheck = true;
```

```
,
```

Spell check dictionary cache

Starting from **v20.1.0.xx**, we have optimized the performance and memory usage of spell checker by adding a static method to initialize the dictionaries with specified cache count.

By default, the spell checker holds only one language dictionary in memory. If you want to hold multiple dictionaries in memory, you need to set the cache limit by using **InitializeDictionaries** method as in the below example.

```
`c#
```

```
List<DictionaryData> spellDictCollection = new List<DictionaryData>();
```

```
string personalDictPath = string.Empty;
```

```
int cacheCount = 2;
```

```
// Initialize dictionaries
```

```
SpellChecker.InitializeDictionaries(spellDictCollection, personalDictPath, cacheCount);
```

```
,
```

If dictionaries are initialized using **InitializeDictionaries** method, then we should use default constructor of the **SpellChecker** to check spelling and get suggestion as in the below example code, it will prevent reinitialization of already loaded dictionaries.

```
`c#
```

```
public string SpellCheck([FromBody] SpellCheckJsonData spellChecker)
```

```
{
```

```
try {
```

```
SpellChecker spellCheck = new SpellChecker();
```



```
spellCheck.GetSuggestions(spellChecker.LanguageID, spellChecker.TexttoCheck,
spellChecker.CheckSpelling, spellChecker.CheckSuggestion, spellChecker.AddWord);
return Newtonsoft.Json.JsonConvert.SerializeObject(spellCheck);
}
catch
{
return "{\"SpellCollection\":[],\"HasSpellingError\":false,\"Suggestions\":null}";
}
}
,
```

Previously on every `SpellChecker.GetSuggestion()` method call, the `.aff` and dictionary data will be parsed to generate suggestion for miss spelled word. But, starting from v20.1.0.xx, the `.aff` and dictionary data will be parsed only for the first time alone while calling `SpellChecker.GetSuggestion()` method.

Add new root word and possible words to dictionary

If you find any root word is missing in the dictionary file, then you can add that new root word and the rule to form the possible words to dictionary file using `AddNewWord` API in the server-side Spell check library.

Note:

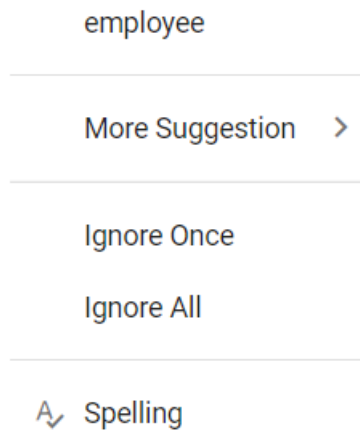
1. The rules are framed automatically using the root word, the possible words and affix file.
2. If you pass null for the parameters `affPath` and `possibleWords`, then it will add a single root word to dictionary.
3. This API is included starting from v20.2.0.xx.

The following code example demonstrates how to add a new root word to the dictionary along with the rule to form the possible words.

```
`c#
SpellChecker spellChecker = new SpellChecker();
// Adds the specified new root word to the dictionary along with the rule to form the possible words.
spellChecker.AddNewWord("en.dic","en.aff", "construct", new string[] { "constructs", "reconstruct",
"constructed", "constructive" });
,
```

Context menu

Right click on error word to open the context menu with spell check options. Please see below screenshot for your reference.



Suggestions

Context menu shows the suggestions for mis-spelled words. By clicking on the required word from suggestion, the error word gets replaced automatically.

Add To Dictionary

Using this option, you can add the current word to the dictionary. So that the spell checker does not consider that word as error in future.

Ignore Once and Ignore All

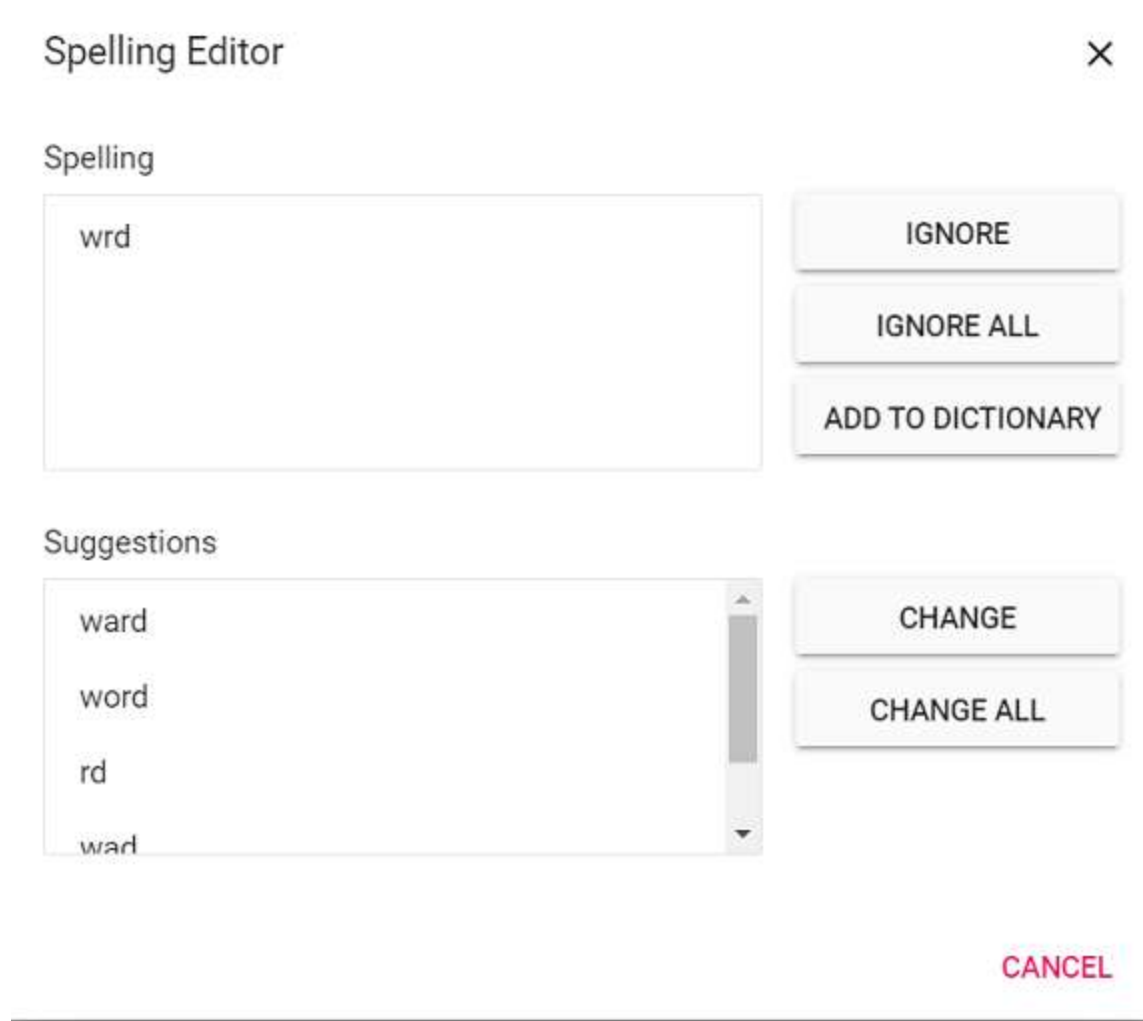
If you do not wish to add the word to dictionary and do not want to show error, use Ignore Once or Ignore All options.

Ignore: ignore only the current occurrence of a word from error.

Ignore All: ignore all occurrence of a word from error in the entire document.

Spelling

Using this option, you can open spell check dialog. Please see below screenshot for your reference.



Global local in ##Platform_Name## Document editor control

Localization

The [Localization](#) library allows you to localize default text content of the DocumentEditor. The Document Editor component has static text on some features (like find & replace, context-menu, dialogs) that can be changed to other cultures (Arabic, Deutsch, French, etc.) by defining the locale value and translation object. Please refer the sample link [RTL](#)

Note: Please refer the [Locale](#).

Document Editor

The following list of properties and its values are used in the Document Editor.

Locale keywords |Text

New | New

Open | Open

Undo | Undo

Redo | Redo

Image | Image

Table | Table

Link | Link

Bookmark | Bookmark

Table of Contents | Table of Contents

HEADING ----- 1 | HEADING ----- 1

HEADING ----- 2 | HEADING ----- 2

HEADING ----- 3 | HEADING ----- 3

Header | Header

Footer | Footer

Page Setup | Page Setup

Page Number | Page Number

Break | Break

Find | Find

Local Clipboard | Local Clipboard

Restrict Editing | Restrict Editing

Upload from computer | Upload from computer

By URL | By URL

Page Break | Page Break

Section Break | Section Break

Header And Footer | Header & Footer

Options | Options

Levels | Levels

Different First Page | Different First Page

Different header and footer for odd and even pages | Different header and footer for odd and even pages.

Different Odd And Even Pages | Different Odd & Even Pages

Different header and footer for first page | Different header and footer for first page.

Position | Position

Header from Top | Header from Top

Footer from Bottom | Footer from Bottom

Distance from top of the page to top of the header | Distance from top of the page to top of the header.

Distance from bottom of the page to bottom of the footer | Distance from bottom of the page to bottom of the footer.

Aspect ratio | Aspect ratio

W | W

H | H

Width | Width

Height | Height

Text | Text

Paragraph | Paragraph

Fill | Fill

Fill color | Fill color

Border Style | Border Style

Outside borders | Outside borders

All borders | All borders

Inside borders | Inside borders

Left border | Left border

Inside vertical border | Inside vertical border

Right border | Right border

Top border | Top border

Inside horizontal border | Inside horizontal border

Bottom border | Bottom border

Border color | Border color

Border width | Border width

Cell | Cell

Merge cells | Merge cells

Insert Or Delete | Insert / Delete

Insert columns to the left | Insert columns to the left

Insert columns to the right | Insert columns to the right

Insert rows above | Insert rows above

Insert rows below | Insert rows below

Delete rows | Delete rows

Delete columns | Delete columns

Cell Margin | Cell Margin

Top | Top

Bottom | Bottom

Left | Left

Right | Right

Align Text | Align Text

Align top | Align top

Align bottom | Align bottom

Align center | Align center

Number of heading or outline levels to be shown in table of contents | Number of heading or outline levels to be shown in table of contents.

Show page numbers | Show page numbers

Show page numbers in table of contents | Show page numbers in table of contents.

Right align page numbers | Right align page numbers

Right align page numbers in table of contents | Right align page numbers in table of contents.

Use hyperlinks | Use hyperlinks

Use hyperlinks instead of page numbers | Use hyperlinks instead of page numbers.

Font | Font

Font Size | Font Size

Font color | Font color

Text highlight color | Text highlight color

Clear all formatting | Clear all formatting

Bold Tooltip | Bold (Ctrl+B)

Italic Tooltip | Italic (Ctrl+I)

Underline Tooltip | Underline (Ctrl+U)

Strikethrough | Strikethrough

Superscript Tooltip | Superscript (Ctrl+Shift++)

Subscript Tooltip | Subscript (Ctrl+=)

Align left Tooltip | Align left (Ctrl+L)

Center Tooltip | Center (Ctrl+E)

Align right Tooltip | Align right (Ctrl+R)

Justify Tooltip | Justify (Ctrl+J)

Decrease indent | Decrease indent

Increase indent | Increase indent

Line spacing | Line spacing

Bullets | Bullets

Numbering | Numbering

Styles | Styles

Manage Styles | Manage Styles

Page | Page

of | of

Fit one page | Fit one page

Spell Check | Spell Check

Underline errors | Underline errors

Fit page width | Fit page width

Update | Update

Cancel | Cancel

Insert | Insert

No Border | No Border

Create a new document | Create a new document.

Open a document | Open a document.

Undo Tooltip | Undo the last operation (Ctrl+Z).

Redo Tooltip | Redo the last operation (Ctrl+Y).

Insert inline picture from a file | Insert inline picture from a file.

Insert a table into the document | Insert a table into the document

Create Hyperlink | Create a link in your document for quick access to web pages and files (Ctrl+K).

Insert a bookmark in a specific place in this document | Insert a bookmark in a specific place in this document.

Provide an overview of your document by adding a table of contents | Provide an overview of your document by adding a table of contents.

Add or edit the header | Add or edit the header.

Add or edit the footer | Add or edit the footer.

Open the page setup dialog | Open the page setup dialog.

Add page numbers | Add page numbers.

Find Text | Find text in the document (Ctrl+F).

Toggle between the internal clipboard and system clipboard | Toggle between the internal clipboard and system clipboard.</br>Access to system clipboard through script is denied due to browsers security policy. Instead, </br> 1. You can enable internal clipboard to cut, copy and paste within the component.</br> 2. You can use the keyboard shortcuts (Ctrl+X, Ctrl+C and Ctrl+V) to cut, copy and paste with system clipboard.

Current Page Number | The current page number in the document. Click or tap to navigate specific page.

Read only | Read only

Protections | Protections

Error in establishing connection with web server | Error in establishing connection with web server

Single | Single

Double | Double

New comment | New comment

Comments | Comments

Print layout | Print layout

Web layout | Web layout

Text Form | Text Form

Check Box | Check Box

DropDown | Drop-Down

Update Fields | Update Fields

Update cross reference fields | Update cross reference fields

Hide properties pane | Hide properties pane

Show properties pane | Show properties pane

[Color Picker](#)

The following list of properties and its values are used in the color picker.

Locale keywords |Text

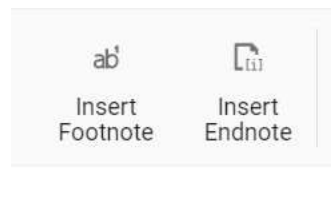
Apply | Apply

Cancel | Cancel

ModeSwitcher | Switch Mode

[Notes in ##Platform_Name## Document editor control](#)

DocumentEditorContainer component provides support for inserting footnotes and endnotes through the in-built toolbar. Refer to the following screenshot.



The Footnotes and endnotes are both ways of adding extra bits of information to your writing outside of the main text. You can use footnotes and endnotes to add side comments to your work or to place other publications like books, articles, or websites.

Insert footnotes

Document Editor exposes an API to insert footnotes at cursor position programmatically or can be inserted to the end of selected text.

```
`ts
```

```
import { DocumentEditorContainer, Toolbar } from '@syncfusion/ej2-documenteditor';
```

```
//Inject require modules.
```

```
DocumentEditorContainer.Inject(Toolbar);
```

```
let container: DocumentEditorContainer = new DocumentEditorContainer({
```

```
enableToolbar: true,
```

```
serviceUrl: 'https://services.syncfusion.com/js/production/api/documenteditor/'
```

```
});
```

```
container.appendTo('#DocumentEditor');
```

```
//Insert footnote in current selection.
```

```
container.documentEditor.editor.insertFootnote();
```

```
,
```

Insert endnotes

Document Editor exposes an API to insert endnotes at cursor position programmatically or can be inserted to the end of selected text.

```
`ts
```

```
import { DocumentEditorContainer, Toolbar } from '@syncfusion/ej2-documenteditor';
```

```
//Inject require modules.
```

```
DocumentEditorContainer.Inject(Toolbar);
```

```
let container: DocumentEditorContainer = new DocumentEditorContainer({
```

```
enableToolbar: true,
```

```
serviceUrl: 'https://services.syncfusion.com/js/production/api/documenteditor/'
```

```
});
```

```
container.appendTo('#DocumentEditor');
```

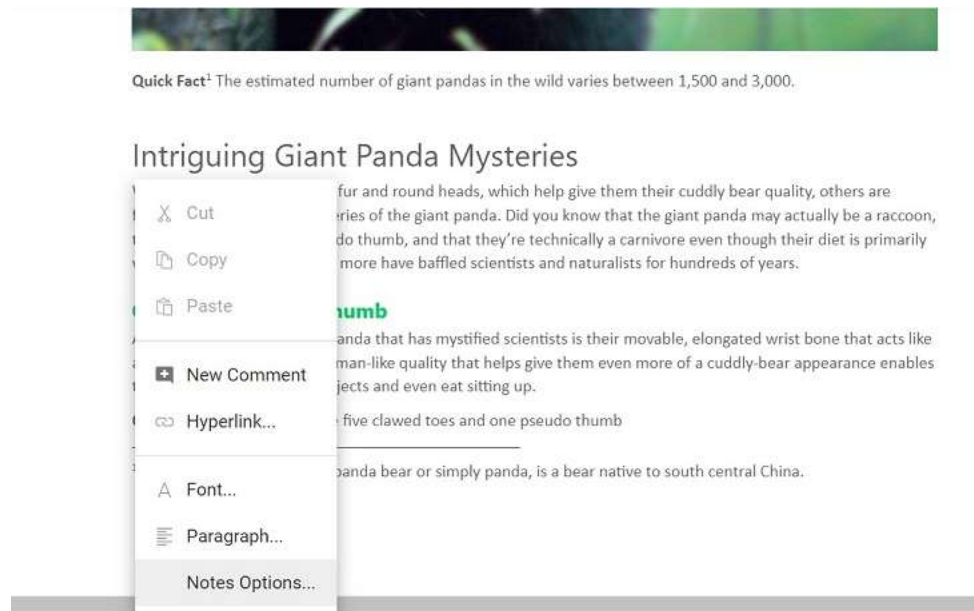
```
//Insert endnote in current selection.
```

```
container.documentEditor.editor.insertEndnote();
```

```
,
```

Update or edit footnotes and endnotes

You can update or edit the footnotes and endnotes using the built-in context menu shown up by right-clicking it. The footnote endnote dialog box popup and you can customize the number format and start at. Refer to the following screenshot.



How To

Override the keyboard shortcuts in `##Platform_Name##` Document editor control

Document Editor triggers the [keyDown](#) event every time when any key is entered and provides an instance of [DocumentEditorKeyDownEventArgs](#). You can use the [isHandled](#) property to override the keyboard shortcut behavior.

Preventing default keyboard shortcut

The following code shows how to prevent the **CTRL + C** keyboard shortcut for copying selected content in Document Editor.

INDEX.TS

```
import { DocumentEditor, Selection, Editor, DocumentEditorKeyDownEventArgs }
from '@syncfusion/ej2-documenteditor';
//Inject require modules.
DocumentEditor.Inject(Selection, Editor)
//Initialize Document Editor.
let documentEditor: DocumentEditor = new DocumentEditor({ enableEditor:
true, isReadOnly: false, height: '370px' });
documentEditor.appendTo('#DocumentEditor');
//Prevent keyboard shortcut inside `keyDown` event.
documentEditor.keyDown = function (args: DocumentEditorKeyDownEventArgs) {
    let keyCode: number = args.event.which || args.event.keyCode;
    let isCtrlKey: boolean = (args.event.ctrlKey || args.event.metaKey) ?
true : ((keyCode === 17) ? true : false);
    //67 is the character code for 'C'
    if (isCtrlKey && keyCode === 67) {
        //To prevent copy operation set isHandled to true
        args.isHandled = true;
    }
}
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
documenteditor/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/fabric.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">

    <div id="DocumentEditor">

    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Override or define the keyboard shortcut

Override or define a new keyboard shortcut behavior instead of preventing the keyboard shortcut.

For example, **Ctrl + S** keyboard shortcut saves the document in SFDT format by default, and there is no behavior for **Ctrl + Alt + S**. The following code demonstrates how to override the **Ctrl + S** shortcut to save a document in DOCX format and define **Ctrl + Alt + S** to save the document in SFDT format.

INDEX.TS

```
import { DocumentEditor, Selection, Editor, DocumentEditorKeyDownEventArgs,
SfdtExport, WordExport } from '@syncfusion/ej2-documenteditor';
//Inject require modules.
DocumentEditor.Inject(Selection, Editor, SfdtExport, WordExport);
//Initialize Document Editor.
let documentEditor: DocumentEditor = new DocumentEditor({ height: '370px',
enableEditor: true, enableSfdtExport: true, enableWordExport: true,
isReadOnly: false });
documentEditor.appendTo('#DocumentEditor');
//Override keyboard shortcut inside `keyDown` event.
documentEditor.keyDown = function (args: DocumentEditorKeyDownEventArgs) {
    let keyCode: number = args.event.which || args.event.keyCode;
    let isCtrlKey: boolean = (args.event.ctrlKey || args.event.metaKey) ?
true : ((keyCode === 17) ? true : false);
    let isAltKey: boolean = args.event.altKey ? args.event.altKey :
((keyCode === 18) ? true : false);
    //83 is the character code for 'S'
    if (isCtrlKey && !isAltKey && keyCode === 83) {
        //To prevent default save operation, set the isHandled property to
true
        args.isHandled = true;
        documentEditor.save('sample', 'Docx');
        args.event.preventDefault();
    } else if (isCtrlKey && isAltKey && keyCode === 83) {
        documentEditor.save('sample', 'Sfdt');
    }
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
documenteditor/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/fabric.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">

        <div id="DocumentEditor">

            </div>
        </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customize context menu in ##Platform_Name## Document editor control

How to customize context menu in Document Editor

Document Editor allows you to add custom option in context menu. It can be achieved by using the [addCustomMenu\(\)](#) method and custom action is defined using the [customContextMenuSelect](#)

Add Custom Option

The following code shows how to add custom option in context menu.

```
`ts
```

```

let documentEditor: DocumentEditor = new DocumentEditor({
isReadOnly: false, serviceUrl: 'https://services.syncfusion.com/js/production/api/documenteditor/'
});
documentEditor.enableAllModules();
documentEditor.appendTo('#DocumentEditor');
//Creating custom menu items
let menuItems: MenuItemModel[] = [
{
text: 'Search In Google',
id: 'searchingoogle',
iconCss: 'e-icons e-de-ctnr-find'
}
];

```

```
//Adding custom options
documentEditor.contextMenu.addCustomMenu(menuItems, false);
//To handle contextmenu select event
documentEditor.customContextMenuSelect = (args: CustomContextMenuEventArgs): void => {
  let item: string = args.id;
  let id: string = documentEditor.element.id;
  switch (item) {
    case id + 'searchingoogle':
      let searchContent: string = documentEditor.selection.text;
      if (!documentEditor.selection.isEmpty && /\S/.test(searchContent)) {
        window.open('http://google.com/search?q=' + searchContent);
      }
      break;
  }
};
`
```

[Customize custom option in context menu](#)

Document Editor allows you to customize the added custom option and also to hide/show default context menu.

[Hide default context menu items](#)

The following code shows how to hide default context menu and add custom option in context menu.

```
`ts
let documentEditor: DocumentEditor = new DocumentEditor({
  isReadOnly: false, serviceUrl: 'https://services.syncfusion.com/js/production/api/documenteditor/'
});
documentEditor.enableAllModules();
documentEditor.appendTo('#DocumentEditor');
//Creating custom menu items
let menuItems: MenuItemModel[] = [
  {
    text: 'Search In Google',
    id: 'searchingoogle',
    iconCss: 'e-icons e-de-ctnr-find'
  }
];
```

//Adding custom options

```
documentEditor.contextMenu.addCustomMenu(menuItems, true);
```

,

[Customize added context menu items](#)

The following code shows how to hide/show added custom option in context menu using the [customContextMenuBeforeOpen](#).

`ts

```
let documentEditor: DocumentEditor = new DocumentEditor({
  isReadOnly: false, serviceUrl: 'https://services.syncfusion.com/js/production/api/documenteditor/'
});
```

```
documentEditor.enableAllModules();
```

```
documentEditor.appendTo('#DocumentEditor');
```

//Creating custom menu items

```
let menuItems: MenuItemModel[] = [
```

```
{
```

```
  text: 'Search In Google',
```

```
  id: 'searchingoogle',
```

```
  iconCss: 'e-icons e-de-ctnr-find'
```

```
});
```

//Adding custom options

```
documentEditor.contextMenu.addCustomMenu(menuItems, false);
```

//To show/hide custom menu item

```
documentEditor.customContextMenuBeforeOpen = (args:
  BeforeOpenCloseCustomContentMenuEventArgs): void => {
```

```
  let search: HTMLElement = document.getElementById(args.ids[0]);
```

```
  search.style.display = 'none';
```

```
  let searchContent: string = documentEditor.selection.text;
```

```
  if ((!documentEditor.selection.isEmpty) && (/S/.test(searchContent))) {
```

```
    search.style.display = 'block';
```

```
  }
```

```
};
```

//To handle contextmenu select event

```
documentEditor.customContextMenuSelect = (args: CustomContentMenuEventArgs): void => {
```

```
  let item: string = args.id;
```

```

let id: string = documentEditor.element.id;
switch (item) {
case id + 'searchingoogle':
let searchContent: string = documentEditor.selection.text;
if (!documentEditor.selection.isEmpty && /\S/.test(searchContent)) {
window.open('http://google.com/search?q=' + searchContent);
}
break;
}
};

```

The following is the output of custom context menu with customization.

INDEX.JS

```

ej.documenteditor.Editor, ej.documenteditor.Selection, ej.documenteditor.Option
nsPane, ej.documenteditor.Search, ej.documenteditor.ContextMenu,
ej.documenteditor.EditorHistory, ej.documenteditor.ImageResizer,
ej.documenteditor.ListDialog, ej.documenteditor.TableDialog,
ej.documenteditor.HyperlinkDialog, ej.documenteditor.ParagraphDialog,
ej.documenteditor.FontDialog, ej.documenteditor.PageSetupDialog,
ej.documenteditor.BookmarkDialog, ej.documenteditor.StyleDialog,
ej.documenteditor.TablePropertiesDialog,
ej.documenteditor.BordersAndShadingDialog,
ej.documenteditor.TableOptionsDialog, ej.documenteditor.CellOptionsDialog,
ej.documenteditor.TableOfContentsDialog
var defaultCheckBoxObj = new ej.buttons.CheckBox({ label: 'Hide Default
Context Menu', change: contextmenuHelper });
defaultCheckBoxObj.appendTo('#default-context-menu');
var positionCheckBoxObj = new ej.buttons.CheckBox({ label: 'Add Custom
option at bottom', change: contextmenuHelper });
positionCheckBoxObj.appendTo('#position-context-menu');
var editor = new ej.documenteditor.DocumentEditor({
  isReadOnly: false,
  serviceUrl:
'https://services.syncfusion.com/js/production/api/documenteditor/'
});
editor.enableAllModules();
editor.appendTo('#DocumentEditor');
// Creating custom menu items
var menuItems = [
  {
    text: 'Search In Google',
    id: 'search_in_google',
    iconCss: 'e-icons e-de-ctnr-find'
  }
];
// Adding custom options
editor.contextMenu.addCustomMenu(menuItems, false);
// To show/hide custom menu item

```



```

editor.customContextMenuBeforeOpen = function(args) {
    var search = document.getElementById(args.ids[0]);
    search.style.display = 'none';
    var searchContent = editor.selection.text;
    if ((!editor.selection.isEmpty) && (/S/.test(searchContent))) {
        search.style.display = 'block';
    }
}
// To handle contextmenu select event
editor.customContextMenuSelect = function(args) {
    var item = args.id;
    var id = editor.element.id;
    switch (item) {
        case id + 'search_in_google':
            var searchContent = editor.selection.text;
            if (!editor.selection.isEmpty && /S/.test(searchContent)) {
                window.open('http://google.com/search?q=' + searchContent);
            }
            break;
    }
}
function contextmenuHelper(args) {
    editor.contextMenu.addCustomMenu(menuItems, defaultCheckBoxObj.checked,
    positionCheckBoxObj.checked);
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
documenteditor/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/fabric.css" rel="stylesheet">

```

```
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <input id="default-context-menu" type="checkbox">
    <input id="position-context-menu" type="checkbox">
    <div id="container">
        <div id="DocumentEditor">
            </div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Customize tool bar in ##Platform_Name## Document editor control

How to customize existing toolbar in DocumentEditorContainer

DocumentEditorContainer allows you to customize(add, show, hide, enable, and disable) existing items in a toolbar.

- Add - New items can be defined by [CustomToolbarItemModel](#) and with existing items in [toolbarItems](#) property. Newly added item click action can be defined in [toolbarclick](#).
- Show, Hide - Existing items can be shown or hidden using the [toolbarItems](#) property. Pre-defined toolbar items are available with [ToolbarItem](#).
- Enable, Disable - Toolbar items can be enabled or disable using [enableItems](#)

```
`ts
```

```
let toolItem: CustomToolbarItemModel = {
    prefixIcon: "e-de-ctnr-lock",
    tooltipText: "Disable Image",
    text: "Disable Image",
    id: "Custom"
};

//Initialize Document Editor Container component with custom toolbar item.
let container: DocumentEditorContainer = new DocumentEditorContainer({
    toolbarItems: [toolItem, 'Undo', 'Redo', 'Separator', 'Image', 'Table', 'Hyperlink', 'Bookmark',
    'Comments', 'TableOfContents', 'Separator', 'Header', 'Footer', 'PageSetup', 'PageNumber', 'Break',
    'Separator', 'Find', 'Separator', 'LocalClipboard', 'RestrictEditing']
});
```

```
//To handle custom toolbar click event.
container.toolbarClick = (args: ClickEventArgs): void => {
  switch (args.item.id) {
    case 'Custom':
      //Disable image toolbar item.
      container.toolbar.enableItems(4, false);
      break;
  }
};
`
```

Note: Default value of `toolbarItems` is ['New', 'Open', 'Separator', 'Undo', 'Redo', 'Separator', 'Image', 'Table', 'Hyperlink', 'Bookmark', 'TableOfContents', 'Separator', 'Header', 'Footer', 'PageSetup', 'PageNumber', 'Break', 'InsertFootnote', 'InsertEndnote', 'Separator', 'Find', 'Separator', 'Comments', 'TrackChanges', 'Separator', 'LocalClipboard', 'RestrictEditing', 'Separator', 'FormFields', 'UpdateFields'].

Change document view in `##Platform_Name##` Document editor control

How to change the document view in DocumentEditor component

DocumentEditor allows you to change the view to web layout and print using the [layoutType](#) property with the supported [LayoutType](#).

```
`ts
let docEdit: DocumentEditor = new DocumentEditor({ layoutType: 'Continuous'});
`
```

Note: Default value of [layoutType](#) in DocumentEditor component is [Pages](#).

How to change the document view in DocumentEditorContainer component

DocumentEditorContainer component allows you to change the view to web layout and print using the [layoutType](#) property with the supported [LayoutType](#).

```
`ts
let container: DocumentEditorContainer = new DocumentEditorContainer({ layoutType: "Continuous" });
`
```

Note: Default value of [layoutType](#) in DocumentEditorContainer component is [Pages](#).

Open default document in `##Platform_Name##` Document editor control

In this article, we are going to see how to open a default document when DocumentEditor & DocumentEditorContainer is initialized.

Opening a default document in DocumentEditor

By default, Document Editor will open blank document. You can use [open](#) API in Document Editor to open the sfdt content.

Document editor have [created](#) event which gets triggered once Document editor control created. So, if you want to open the document by default, you can use [open](#) and [created](#) API.

The following example illustrates how to open the default SFDT content once Document editor control gets created.

INDEX.JS

```
var documenteditor = new ej.documenteditor.DocumentEditor({
  isReadOnly: false, serviceUrl:
  'https://services.syncfusion.com/js/production/api/documenteditor/'
});
documenteditor.enableAllModules();
onCreated = function () {
  var data =
  '{"sections":[{"sectionFormat":{"pageWidth":612,"pageHeight":792,"leftMargin":72,"rightMargin":72,"topMargin":72,"bottomMargin":72,"differentFirstPage":false,"differentOddAndEvenPages":false,"headerDistance":36,"footerDistance":36,"bidi":false},"blocks":[{"paragraphFormat":{"afterSpacing":30,"styleName":"Heading 1","listFormat":{},"characterFormat":{},"inlines":[{"characterFormat":{},"text":"Adventure Works Cycles"}]}],"headersFooters":{"header":{"blocks":[{"paragraphFormat":{"listFormat":{},"characterFormat":{},"inlines":[]}}],"footer":{"blocks":[{"paragraphFormat":{"listFormat":{},"characterFormat":{},"inlines":[]}}]},"characterFormat":{"bold":false,"italic":false,"fontSize":11,"fontFamily":"Calibri","underline":"None","strikethrough":"None","baselineAlignment":"Normal","highlightColor":"NoColor","fontColor":"empty","fontSizeBidi":11,"fontFamilyBidi":"Calibri","allCaps":false},"paragraphFormat":{"leftIndent":0,"rightIndent":0,"firstLineIndent":0,"textAlignment":"Left","beforeSpacing":0,"afterSpacing":0,"lineSpacing":1.0791666507720947,"lineSpacingType":"Multiple","listFormat":{},"bidi":false},"defaultTabWidth":36,"trackChanges":false,"enforcement":false,"hashValue":"","saltValue":"","formatting":false,"protectionType":"NoProtection","dontUseHTMLParagraphAutoSpacing":false,"formFieldShading":true,"styles":[{"name":"Normal","type":"Paragraph","paragraphFormat":{"lineSpacing":1.149999976158142,"lineSpacingType":"Multiple","listFormat":{},"characterFormat":{"fontFamily":"Calibri"},"next":"Normal"}, {"name":"Default Paragraph Font","type":"Character","characterFormat":{}}, {"name":"Heading 1 Char","type":"Character","characterFormat":{"fontSize":16,"fontFamily":"Calibri Light","fontColor":"#2F5496"},"basedOn":"Default Paragraph Font"}, {"name":"Heading 1","type":"Paragraph","paragraphFormat":{"beforeSpacing":12,"afterSpacing":0,"outlineLevel":"Level1","listFormat":{},"characterFormat":{"fontSize":16,"fontFamily":"Calibri Light","fontColor":"#2F5496"},"basedOn":"Normal","link":"Heading 1 Char","next":"Normal"}, {"name":"Heading 2 Char","type":"Character","characterFormat":{"fontSize":13,"fontFamily":"Calibri Light","fontColor":"#2F5496"},"basedOn":"Default Paragraph Font"}, {"name":"Heading 2","type":"Paragraph","paragraphFormat":{"beforeSpacing":2,"afterSpacing":6,"outlineLevel":"Level2","listFormat":{},"characterFormat":{"fontSize":13,"fontFamily":"Calibri Light","fontColor":"#2F5496"},"basedOn":"Normal","link":"Heading 2 Char","next":"Normal"}, {"name":"Heading 3","type":"Paragraph","paragraphFormat":{"leftIndent":0,"rightIndent":0,"firstLineIndent":0,"textAlignment":"Left","beforeSpacing":2,"afterSpacing":0,"lineSpacing":1.0791666507720947,"lineSpacingType":"Multiple","outlineLevel":}
```

```

Level3", "listFormat": {}, "characterFormat": { "fontSize": 12, "fontFamily": "Calibri Light", "fontColor": "#1F3763", "basedOn": "Normal", "link": "Heading 3 Char", "next": "Normal" }, { "name": "Heading 3 Char", "type": "Character", "characterFormat": { "fontSize": 12, "fontFamily": "Calibri Light", "fontColor": "#1F3763", "basedOn": "Default Paragraph Font" }, { "name": "Heading 4", "type": "Paragraph", "paragraphFormat": { "leftIndent": 0, "rightIndent": 0, "firstLineIndent": 0, "textAlignment": "Left", "beforeSpacing": 2, "afterSpacing": 0, "lineSpacing": 1.0791666507720947, "lineSpacingType": "Multiple", "outlineLevel": "Level4", "listFormat": {}, "characterFormat": { "italic": true, "fontFamily": "Calibri Light", "fontColor": "#2F5496", "basedOn": "Normal", "link": "Heading 4 Char", "next": "Normal" }, { "name": "Heading 4 Char", "type": "Character", "characterFormat": { "italic": true, "fontFamily": "Calibri Light", "fontColor": "#2F5496", "basedOn": "Default Paragraph Font" }, { "name": "Heading 5", "type": "Paragraph", "paragraphFormat": { "leftIndent": 0, "rightIndent": 0, "firstLineIndent": 0, "textAlignment": "Left", "beforeSpacing": 2, "afterSpacing": 0, "lineSpacing": 1.0791666507720947, "lineSpacingType": "Multiple", "outlineLevel": "Level5", "listFormat": {}, "characterFormat": { "fontFamily": "Calibri Light", "fontColor": "#2F5496", "basedOn": "Normal", "link": "Heading 5 Char", "next": "Normal" }, { "name": "Heading 5 Char", "type": "Character", "characterFormat": { "fontFamily": "Calibri Light", "fontColor": "#2F5496", "basedOn": "Default Paragraph Font" }, { "name": "Heading 6", "type": "Paragraph", "paragraphFormat": { "leftIndent": 0, "rightIndent": 0, "firstLineIndent": 0, "textAlignment": "Left", "beforeSpacing": 2, "afterSpacing": 0, "lineSpacing": 1.0791666507720947, "lineSpacingType": "Multiple", "outlineLevel": "Level6", "listFormat": {}, "characterFormat": { "fontFamily": "Calibri Light", "fontColor": "#1F3763", "basedOn": "Normal", "link": "Heading 6 Char", "next": "Normal" }, { "name": "Heading 6 Char", "type": "Character", "characterFormat": { "fontFamily": "Calibri Light", "fontColor": "#1F3763", "basedOn": "Default Paragraph Font" } } ], "abstractLists": [], "comments": [], "revisions": [], "customXml1": [] }';
    documenteditor.open(data);
};
documenteditor.appendTo('#DocumentEditor');
documenteditor.addEventListener("created", onCreate());

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-documenteditor/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/fabric.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">

        <div id="DocumentEditor" style="height:420px">

            </div>
        </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Opening a default document in DocumentEditorContainer

By default, Document Editor Container will open a blank document. You can use [open](#) API in Document Editor to open the SFDT content.

Document editor Container have [created](#) event which gets triggered once Document editor container control created. So, if you want to open the document by default, you can use [open](#) and [created](#) API.

INDEX.JS

```

var documenteditorcontainer = new
ej.documenteditor.DocumentEditorContainer({
    isReadOnly: false
});
ej.documenteditor.DocumentEditorContainer.Inject(
    ej.documenteditor.Toolbar
);
onCreated = function() {
    var data =

    '{"sections":[{"sectionFormat":{"pageWidth":612,"pageHeight":792,"leftMargin":72,"rightMargin":72,"topMargin":72,"bottomMargin":72,"differentFirstPage":false,"differentOddAndEvenPages":false,"headerDistance":36,"footerDistance":

```

```

36,"bidi":false},"blocks":[{"paragraphFormat":{"afterSpacing":30,"styleName":
:"Heading
1","listFormat":{},"characterFormat":{},"inlines":[{"characterFormat":{},"t
ext":"Adventure Works
Cycles"}]}]},"headersFooters":{"header":{"blocks":[{"paragraphFormat":{"listF
ormat":{},"characterFormat":{},"inlines":[]}}],"footer":{"blocks":[{"paragr
aphFormat":{"listFormat":{},"characterFormat":{},"inlines":[]}}]},"charac
terFormat":{"bold":false,"italic":false,"fontSize":11,"fontFamily":"Calibri"
,"underline":"None","strikethrough":"None","baselineAlignment":"Normal","hig
hlightColor":"NoColor","fontColor":"empty","fontSizeBidi":11,"fontFamilyBidi
":"Calibri","allCaps":false},"paragraphFormat":{"leftIndent":0,"rightIndent"
:0,"firstLineIndent":0,"textAlignment":"Left","beforeSpacing":0,"afterSpacin
g":0,"lineSpacing":1.0791666507720947,"lineSpacingType":"Multiple","listForm
at":{},"bidi":false},"defaultTabWidth":36,"trackChanges":false,"enforcement"
:false,"hashValue":"","saltValue":"","formatting":false,"protectionType":"No
Protection","dontUseHTMLParagraphAutoSpacing":false,"formFieldShading":true,
"styles":[{"name":"Normal","type":"Paragraph","paragraphFormat":{"lineSpacin
g":1.149999976158142,"lineSpacingType":"Multiple","listFormat":{},"characte
rFormat":{"fontFamily":"Calibri"},"next":"Normal"},{"name":"Default
Paragraph Font","type":"Character","characterFormat":{}},{name":"Heading 1
Char","type":"Character","characterFormat":{"fontSize":16,"fontFamily":"Cali
bri Light","fontColor":"#2F5496"},"basedOn":"Default Paragraph
Font"},{"name":"Heading
1","type":"Paragraph","paragraphFormat":{"beforeSpacing":12,"afterSpacing":0
,"outlineLevel":"Level1","listFormat":{},"characterFormat":{"fontSize":16,"
fontFamily":"Calibri
Light","fontColor":"#2F5496"},"basedOn":"Normal","link":"Heading 1
Char","next":"Normal"},{"name":"Heading 2
Char","type":"Character","characterFormat":{"fontSize":13,"fontFamily":"Cali
bri Light","fontColor":"#2F5496"},"basedOn":"Default Paragraph
Font"},{"name":"Heading
2","type":"Paragraph","paragraphFormat":{"beforeSpacing":2,"afterSpacing":6,
"outlineLevel":"Level2","listFormat":{},"characterFormat":{"fontSize":13,"f
ontFamily":"Calibri
Light","fontColor":"#2F5496"},"basedOn":"Normal","link":"Heading 2
Char","next":"Normal"},{"name":"Heading
3","type":"Paragraph","paragraphFormat":{"leftIndent":0,"rightIndent":0,"fir
stLineIndent":0,"textAlignment":"Left","beforeSpacing":2,"afterSpacing":0,"l
ineSpacing":1.0791666507720947,"lineSpacingType":"Multiple","outlineLevel":"
Level3","listFormat":{},"characterFormat":{"fontSize":12,"fontFamily":"Cali
bri Light","fontColor":"#1F3763"},"basedOn":"Normal","link":"Heading 3
Char","next":"Normal"},{"name":"Heading 3
Char","type":"Character","characterFormat":{"fontSize":12,"fontFamily":"Cali
bri Light","fontColor":"#1F3763"},"basedOn":"Default Paragraph
Font"},{"name":"Heading
4","type":"Paragraph","paragraphFormat":{"leftIndent":0,"rightIndent":0,"fir
stLineIndent":0,"textAlignment":"Left","beforeSpacing":2,"afterSpacing":0,"l
ineSpacing":1.0791666507720947,"lineSpacingType":"Multiple","outlineLevel":"
Level4","listFormat":{},"characterFormat":{"italic":true,"fontFamily":"Cali
bri Light","fontColor":"#2F5496"},"basedOn":"Normal","link":"Heading 4
Char","next":"Normal"},{"name":"Heading 4
Char","type":"Character","characterFormat":{"italic":true,"fontFamily":"Cali
bri Light","fontColor":"#2F5496"},"basedOn":"Default Paragraph
Font"},{"name":"Heading
5","type":"Paragraph","paragraphFormat":{"leftIndent":0,"rightIndent":0,"fir
stLineIndent":0,"textAlignment":"Left","beforeSpacing":2,"afterSpacing":0,"l
ineSpacing":1.0791666507720947,"lineSpacingType":"Multiple","outlineLevel":"

```

```

Level5", "listFormat": {}, "characterFormat": {"fontFamily": "Calibri
Light", "fontColor": "#2F5496"}, "basedOn": "Normal", "link": "Heading 5
Char", "next": "Normal"}, {"name": "Heading 5
Char", "type": "Character", "characterFormat": {"fontFamily": "Calibri
Light", "fontColor": "#2F5496"}, "basedOn": "Default Paragraph
Font"}, {"name": "Heading
6", "type": "Paragraph", "paragraphFormat": {"leftIndent": 0, "rightIndent": 0, "fir
stLineIndent": 0, "textAlignment": "Left", "beforeSpacing": 2, "afterSpacing": 0, "l
ineSpacing": 1.0791666507720947, "lineSpacingType": "Multiple", "outlineLevel": "
Level6", "listFormat": {}, "characterFormat": {"fontFamily": "Calibri
Light", "fontColor": "#1F3763"}, "basedOn": "Normal", "link": "Heading 6
Char", "next": "Normal"}, {"name": "Heading 6
Char", "type": "Character", "characterFormat": {"fontFamily": "Calibri
Light", "fontColor": "#1F3763"}, "basedOn": "Default Paragraph
Font"}], "lists": [], "abstractLists": [], "comments": [], "revisions": [], "customXm
l": []}';
    documenteditorcontainer.documentEditor.open(data);
};
documenteditorcontainer.appendTo("#DocumentEditor");
documenteditorcontainer.addEventListener("created", onCreate());

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
documenteditor/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/fabric.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```



```

    <div id="container">

        <div id="DocumentEditor" style="height:420px">

            </div>
        </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Read only default in ##Platform_Name## Document editor control

In this article, we are going to see how to open a document in read only mode by default in DocumentEditor & DocumentEditorContainer.

Opening a document in read only mode by default in DocumentEditor

INDEX.JS

```

var documenteditor = new ej.documenteditor.DocumentEditor({
    isReadOnly: false
});
documenteditor.serviceUrl =
'https://services.syncfusion.com/js/production/api/documenteditor/';
documenteditor.documentChange = () => {
    documenteditor.isReadOnly=true;
};
documenteditor.enableAllModules();
documenteditor.appendTo('#DocumentEditor');
let data =
'{"sections":[{"sectionFormat":{"pageWidth":612,"pageHeight":792,"leftMargin":72,"rightMargin":72,"topMargin":72,"bottomMargin":72,"differentFirstPage":false,"differentOddAndEvenPages":false,"headerDistance":36,"footerDistance":36,"bidi":false},"blocks":[{"paragraphFormat":{"afterSpacing":30,"styleName":"Heading1","listFormat":{},"characterFormat":{},"inlines":[{"characterFormat":{},"text":"Adventure Works Cycles"}]}]}],"headersFooters":{"header":{"blocks":[{"paragraphFormat":{"listFormat":{},"characterFormat":{},"inlines":[]}}],"footer":{"blocks":[{"paragraphFormat":{"listFormat":{},"characterFormat":{},"inlines":[]}}]}],"characterFormat":{"bold":false,"italic":false,"fontSize":11,"fontFamily":"Calibri","underline":"None","strikethrough":"None","baselineAlignment":"Normal","highlightColor":"NoColor","fontColor":"empty","fontSizeBidi":11,"fontFamilyBidi":"Calibri","allCaps":false},"paragraphFormat":{"leftIndent":0,"rightIndent":0,"firstLineIndent":0,"textAlignment":"Left","beforeSpacing":0,"afterSpacing":0,"lineSpacing":1.0791666507720947,"lineSpacingType":"Multiple","listFormat":{},"bidi":false},"defaultTabWidth":36,"trackChanges":false,"enforcement":false,"hashValue":"","saltValue":"","formatting":false,"protectionType":"NoProtection","dontUseHTMLParagraphAutoSpacing":false,"formFieldShading":true,"styles":[{"name":"Normal","type":"Paragraph","paragraphFormat":{"lineSpacing":1.149999976158142,"lineSpacingType":"Multiple","listFormat":{},"characterFormat":{"fontFamily":"Calibri"},"next":"Normal"}, {"name":"Default

```

```

Paragraph Font", "type": "Character", "characterFormat": {}, {"name": "Heading 1
Char", "type": "Character", "characterFormat": {"fontSize": 16, "fontFamily": "Cali
bri Light", "fontColor": "#2F5496"}, "basedOn": "Default Paragraph
Font"}, {"name": "Heading
1", "type": "Paragraph", "paragraphFormat": {"beforeSpacing": 12, "afterSpacing": 0
, "outlineLevel": "Level1", "listFormat": {}}, "characterFormat": {"fontSize": 16, "
fontFamily": "Calibri
Light", "fontColor": "#2F5496"}, "basedOn": "Normal", "link": "Heading 1
Char", "next": "Normal"}, {"name": "Heading 2
Char", "type": "Character", "characterFormat": {"fontSize": 13, "fontFamily": "Cali
bri Light", "fontColor": "#2F5496"}, "basedOn": "Default Paragraph
Font"}, {"name": "Heading
2", "type": "Paragraph", "paragraphFormat": {"beforeSpacing": 2, "afterSpacing": 6,
"outlineLevel": "Level2", "listFormat": {}}, "characterFormat": {"fontSize": 13, "f
ontFamily": "Calibri
Light", "fontColor": "#2F5496"}, "basedOn": "Normal", "link": "Heading 2
Char", "next": "Normal"}, {"name": "Heading
3", "type": "Paragraph", "paragraphFormat": {"leftIndent": 0, "rightIndent": 0, "fir
stLineIndent": 0, "textAlignment": "Left", "beforeSpacing": 2, "afterSpacing": 0, "l
ineSpacing": 1.0791666507720947, "lineSpacingType": "Multiple", "outlineLevel": "
Level3", "listFormat": {}}, "characterFormat": {"fontSize": 12, "fontFamily": "Cali
bri Light", "fontColor": "#1F3763"}, "basedOn": "Normal", "link": "Heading 3
Char", "next": "Normal"}, {"name": "Heading 3
Char", "type": "Character", "characterFormat": {"fontSize": 12, "fontFamily": "Cali
bri Light", "fontColor": "#1F3763"}, "basedOn": "Default Paragraph
Font"}, {"name": "Heading
4", "type": "Paragraph", "paragraphFormat": {"leftIndent": 0, "rightIndent": 0, "fir
stLineIndent": 0, "textAlignment": "Left", "beforeSpacing": 2, "afterSpacing": 0, "l
ineSpacing": 1.0791666507720947, "lineSpacingType": "Multiple", "outlineLevel": "
Level4", "listFormat": {}}, "characterFormat": {"italic": true, "fontFamily": "Cali
bri Light", "fontColor": "#2F5496"}, "basedOn": "Normal", "link": "Heading 4
Char", "next": "Normal"}, {"name": "Heading 4
Char", "type": "Character", "characterFormat": {"italic": true, "fontFamily": "Cali
bri Light", "fontColor": "#2F5496"}, "basedOn": "Default Paragraph
Font"}, {"name": "Heading
5", "type": "Paragraph", "paragraphFormat": {"leftIndent": 0, "rightIndent": 0, "fir
stLineIndent": 0, "textAlignment": "Left", "beforeSpacing": 2, "afterSpacing": 0, "l
ineSpacing": 1.0791666507720947, "lineSpacingType": "Multiple", "outlineLevel": "
Level5", "listFormat": {}}, "characterFormat": {"fontFamily": "Calibri
Light", "fontColor": "#2F5496"}, "basedOn": "Normal", "link": "Heading 5
Char", "next": "Normal"}, {"name": "Heading 5
Char", "type": "Character", "characterFormat": {"fontFamily": "Calibri
Light", "fontColor": "#2F5496"}, "basedOn": "Default Paragraph
Font"}, {"name": "Heading
6", "type": "Paragraph", "paragraphFormat": {"leftIndent": 0, "rightIndent": 0, "fir
stLineIndent": 0, "textAlignment": "Left", "beforeSpacing": 2, "afterSpacing": 0, "l
ineSpacing": 1.0791666507720947, "lineSpacingType": "Multiple", "outlineLevel": "
Level6", "listFormat": {}}, "characterFormat": {"fontFamily": "Calibri
Light", "fontColor": "#1F3763"}, "basedOn": "Normal", "link": "Heading 6
Char", "next": "Normal"}, {"name": "Heading 6
Char", "type": "Character", "characterFormat": {"fontFamily": "Calibri
Light", "fontColor": "#1F3763"}, "basedOn": "Default Paragraph
Font"}], "lists": [], "abstractLists": [], "comments": [], "revisions": [], "customXm
l": []}';
// Open the default document
documenteditor.open(data);

```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
documenteditor/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/fabric.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>

<body>

  <div id="container">

    <div id="DocumentEditor" style="height:420px">

    </div>
  </div>

  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Opening a document in ready only mode by default in DocumentEditorContainer

INDEX.JS

```
var documenteditorContainer = new
ej.documenteditor.DocumentEditorContainer({ enableToolbar: true });

ej.documenteditor.DocumentEditorContainer.Inject(ej.documenteditor.Toolbar);
documenteditorContainer.serviceUrl =
'https://services.syncfusion.com/js/production/api/documenteditor/';
//DocumentEditorContainer control rendering starts
documenteditorContainer.documentChange = () => {
    documenteditorContainer.restrictEditing=true;
};
documenteditorContainer.appendTo('#DocumentEditor');
let data=
{"sections":[{"sectionFormat":{"pageWidth":612,"pageHeight":792,"leftMargin":72,"rightMargin":72,"topMargin":72,"bottomMargin":72,"differentFirstPage":false,"differentOddAndEvenPages":false,"headerDistance":36,"footerDistance":36,"bidi":false},"blocks":[{"paragraphFormat":{"afterSpacing":30,"styleName":"Heading 1","listFormat":{},"characterFormat":{},"inlines":[{"characterFormat":{},"text":"Adventure Works Cycles"}]}]},"headersFooters":{"header":{"blocks":[{"paragraphFormat":{"listFormat":{},"characterFormat":{},"inlines":[]}}],"footer":{"blocks":[{"paragraphFormat":{"listFormat":{},"characterFormat":{},"inlines":[]}}]},"characterFormat":{"bold":false,"italic":false,"fontSize":11,"fontFamily":"Calibri","underline":"None","strikethrough":"None","baselineAlignment":"Normal","highlightColor":"NoColor","fontColor":"empty","fontSizeBidi":11,"fontFamilyBidi":"Calibri","allCaps":false},"paragraphFormat":{"leftIndent":0,"rightIndent":0,"firstLineIndent":0,"textAlignment":"Left","beforeSpacing":0,"afterSpacing":0,"lineSpacing":1.0791666507720947,"lineSpacingType":"Multiple","listFormat":{},"bidi":false},"defaultTabWidth":36,"trackChanges":false,"enforcement":false,"hashValue":"","saltValue":"","formatting":false,"protectionType":"NoProtection","dontUseHTMLParagraphAutoSpacing":false,"formFieldShading":true,"styles":[{"name":"Normal","type":"Paragraph","paragraphFormat":{"lineSpacing":1.149999976158142,"lineSpacingType":"Multiple","listFormat":{},"characterFormat":{"fontFamily":"Calibri"},"next":"Normal"},{"name":"Default Paragraph Font","type":"Character","characterFormat":{}},{name":"Heading 1 Char","type":"Character","characterFormat":{"fontSize":16,"fontFamily":"Calibri Light","fontColor":"#2F5496"},"basedOn":"Default Paragraph Font"},{"name":"Heading 1","type":"Paragraph","paragraphFormat":{"beforeSpacing":12,"afterSpacing":0,"outlineLevel":"Level1","listFormat":{},"characterFormat":{"fontSize":16,"fontFamily":"Calibri Light","fontColor":"#2F5496"},"basedOn":"Normal","link":"Heading 1 Char","next":"Normal"},{"name":"Heading 2 Char","type":"Character","characterFormat":{"fontSize":13,"fontFamily":"Calibri Light","fontColor":"#2F5496"},"basedOn":"Default Paragraph Font"},{"name":"Heading 2","type":"Paragraph","paragraphFormat":{"beforeSpacing":2,"afterSpacing":6,"outlineLevel":"Level2","listFormat":{},"characterFormat":{"fontSize":13,"fontFamily":"Calibri Light","fontColor":"#2F5496"},"basedOn":"Normal","link":"Heading 2 Char","next":"Normal"},{"name":"Heading 3","type":"Paragraph","paragraphFormat":{"leftIndent":0,"rightIndent":0,"firstLineIndent":0,"textAlignment":"Left","beforeSpacing":2,"afterSpacing":0,"lineSpacing":1.0791666507720947,"lineSpacingType":"Multiple","outlineLevel":}
```

```

Level3", "listFormat": {}, "characterFormat": { "fontSize": 12, "fontFamily": "Calibri Light", "fontColor": "#1F3763", "basedOn": "Normal", "link": "Heading 3 Char", "next": "Normal" }, { "name": "Heading 3 Char", "type": "Character", "characterFormat": { "fontSize": 12, "fontFamily": "Calibri Light", "fontColor": "#1F3763", "basedOn": "Default Paragraph Font" }, { "name": "Heading 4", "type": "Paragraph", "paragraphFormat": { "leftIndent": 0, "rightIndent": 0, "firstLineIndent": 0, "textAlignment": "Left", "beforeSpacing": 2, "afterSpacing": 0, "lineSpacing": 1.0791666507720947, "lineSpacingType": "Multiple", "outlineLevel": "Level4", "listFormat": {}, "characterFormat": { "italic": true, "fontFamily": "Calibri Light", "fontColor": "#2F5496", "basedOn": "Normal", "link": "Heading 4 Char", "next": "Normal" }, { "name": "Heading 4 Char", "type": "Character", "characterFormat": { "italic": true, "fontFamily": "Calibri Light", "fontColor": "#2F5496", "basedOn": "Default Paragraph Font" }, { "name": "Heading 5", "type": "Paragraph", "paragraphFormat": { "leftIndent": 0, "rightIndent": 0, "firstLineIndent": 0, "textAlignment": "Left", "beforeSpacing": 2, "afterSpacing": 0, "lineSpacing": 1.0791666507720947, "lineSpacingType": "Multiple", "outlineLevel": "Level5", "listFormat": {}, "characterFormat": { "fontFamily": "Calibri Light", "fontColor": "#2F5496", "basedOn": "Normal", "link": "Heading 5 Char", "next": "Normal" }, { "name": "Heading 5 Char", "type": "Character", "characterFormat": { "fontFamily": "Calibri Light", "fontColor": "#2F5496", "basedOn": "Default Paragraph Font" }, { "name": "Heading 6", "type": "Paragraph", "paragraphFormat": { "leftIndent": 0, "rightIndent": 0, "firstLineIndent": 0, "textAlignment": "Left", "beforeSpacing": 2, "afterSpacing": 0, "lineSpacing": 1.0791666507720947, "lineSpacingType": "Multiple", "outlineLevel": "Level6", "listFormat": {}, "characterFormat": { "fontFamily": "Calibri Light", "fontColor": "#1F3763", "basedOn": "Normal", "link": "Heading 6 Char", "next": "Normal" }, { "name": "Heading 6 Char", "type": "Character", "characterFormat": { "fontFamily": "Calibri Light", "fontColor": "#1F3763", "basedOn": "Default Paragraph Font" } }], "lists": [], "abstractLists": [], "comments": [], "revisions": [], "customXml": [];
// Open the default document
documenteditorContainer.documentEditor.open(JSON.stringify(data));

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-documenteditor/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/fabric.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>

<body>

  <div id="container">

    <div id="DocumentEditor" style="height:420px">

      </div>
    </div>

<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Note: You can use the [restrictEditing](#) in DocumentEditorContainer and [isReadOnly](#) in DocumentEditor based on your requirement to change component to read only mode.

Open document by address in ##Platform_Name## Document editor control

How to open a document from URL in DocumentEditor

In this article, we are going to see how to open a document from URL in DocumentEditor

please refer below example for client-side code

```
`ts
```

```
//Initialize Document Editor Container component.
```

```
let container: DocumentEditorContainer = new DocumentEditorContainer();
```

```
container.appendTo('#DocumentEditorContainer');
```

```
document.getElementById('import').addEventListener('click', () => {
```

```
let http: XMLHttpRequest = new XMLHttpRequest();
```

```
//add your url in which you want to open document inside the ""
```

```
let content = { fileUrl: "" };
```

```
let baseUrl: string = "/api/documenteditor/ImportFileURL";
http.open("POST", baseUrl, true);
http.setRequestHeader("Content-Type", "application/json;charset=UTF-8");
http.onreadystatechange = () => {
    if (http.readyState === 4) {
        if (http.status === 200 || http.status === 304) {
            //open the SFDT text in Document Editor
            container.documentEditor.open(http.responseText);
        }
    }
};
http.send(JSON.stringify(content));
});
`
```

please refer below example for server-side code

```
`c#
[AcceptVerbs("Post")]
public string ImportFileURL([FromBody]FileUrlInfo param)
{
    try {
        using(WebClient client = new WebClient())
        {
            MemoryStream stream = new MemoryStream(client.DownloadData(param.fileUrl));
            WordDocument document = WordDocument.Load(stream, FormatType.Docx);
            string json = Newtonsoft.Json.JsonConvert.SerializeObject(document);
            document.Dispose();
            stream.Dispose();
            return json;
        }
    }
    catch (Exception) {
        return "";
    }
}
```

```

}
public class FileInfo {
public string fileUrl { get; set; }
public string Content { get; set; }
}
`

```

Deploy document editor component for mobile in `##Platform_Name##` Document editor control
[Document editor component for Mobile](#)

At present, Document editor component is not responsive for mobile, and we haven't ensured the editing functionalities in mobile browsers. Whereas it works properly as a document viewer in mobile browsers.

Hence, it is recommended to switch the Document editor component as read-only in mobile browsers. Also, invoke [fitPage](#) method with [FitPageWidth](#) parameter in document change event, such as to display one full page by adjusting the zoom factor.

The following example code illustrates how to deploy Document Editor component for Mobile.

```

`ts
//Initialize Document Editor Container component.
import { DocumentEditorContainer, Toolbar } from '@syncfusion/ej2-documenteditor';
DocumentEditorContainer.Inject(Toolbar);
let container: DocumentEditorContainer = new DocumentEditorContainer({
enableToolbar: true, height: '590px'
});
container.serviceUrl = 'https://services.syncfusion.com/js/production/api/documenteditor/';
container.appendTo('#DocumentEditor');
//To detect the device
let isMobileDevice: bool = /Android|Windows Phone|webOS/i.test(navigator.userAgent);
container.documentChange = () => {
if (isMobileDevice) {
container.restrictEditing = true;
setTimeout(() => {
container.documentEditor.fitPage("FitPageWidth");
}, 50);
}
else {
container.restrictEditing = false;
}
}

```



```

}
}
`

```

You can download the complete working example from this [GitHub location](#)

Note: You can use the [restrictEditing](#) in DocumentEditorContainer and [isReadOnly](#) in DocumentEditor based on your requirement to change component to read only mode.

Disable optimized text measuring in `Platform_Name` Document editor control

Starting from v19.3.0.x, the accuracy of text size measurements in Document editor is improved such as to match Microsoft Word pagination for most Word documents. This improvement is included as default behavior along with an optional API [enableOptimizedTextMeasuring](#) in Document editor settings.

If you want the Document editor component to retain the document pagination (display page-by-page) behavior like v19.2.0.x and older versions. Then you can disable this optimized text measuring improvement, by setting `false` to [enableOptimizedTextMeasuring](#) property of JavaScript Document Editor component.

Disable optimized text measuring in `DocumentEditorContainer` instance

The following example code illustrates how to disable optimized text measuring improvement in `DocumentEditorContainer` instance.

```

`ts
import { DocumentEditorContainer, Toolbar } from '@syncfusion/ej2-documenteditor';
DocumentEditorContainer.Inject(Toolbar);

let container: DocumentEditorContainer = new DocumentEditorContainer({ enableToolbar: true, height:
'590px' });

// Disable optimized text measuring improvement
container.documentEditorSettings = { enableOptimizedTextMeasuring: false };
container.serviceUrl = 'https://services.syncfusion.com/js/production/api/documenteditor/';
container.appendTo('#container');
`

```

Disable optimized text measuring in `DocumentEditor` instance

The following example code illustrates how to disable optimized text measuring improvement in `DocumentEditor` instance.

```

`ts
import { DocumentEditor } from '@syncfusion/ej2-documenteditor';

let documenteditor: DocumentEditor = new DocumentEditor({ isReadOnly: false, height: '370px',
serviceUrl: 'https://services.syncfusion.com/js/production/api/documenteditor/' });

documenteditor.enableAllModules();

// Disable optimized text measuring improvement
documenteditor.documentEditorSettings = { enableOptimizedTextMeasuring: false };

```

```
documenteditor.appendTo('#DocumentEditor');
`
```

Get the selected content in `##Platform_Name##` Document editor control

You can get the selected content from the JavaScript Document Editor component as plain text and SFDT (rich text).

Get the selected content as plain text

You can use [text](#) API to get the selected content as plain text from JavaScript Document Editor component.

The following example code illustrates how to add search in google option in context menu for the selected text.

```
`ts
import { DocumentEditorContainer, Toolbar, CustomContentMenuEventArgs } from '@syncfusion/ej2-
documenteditor';

import { MenuItemModel } from '@syncfusion/ej2-navigations';

DocumentEditorContainer.Inject(Toolbar);

let container: DocumentEditorContainer = new DocumentEditorContainer({ enableToolbar: true, height:
'590px' });

container.serviceUrl = 'https://services.syncfusion.com/js/production/api/documenteditor/';

container.appendTo('#container');

//Creating custom menu items
let menuItems: MenuItemModel[] = [
{
text: 'Search In Google',
id: 'searchingoogle',
iconCss: 'e-icons e-de-ctnr-find'
}
];

//Adding custom options
container.documentEditor.contextMenu.addCustomMenu(menuItems, false);

//To handle contextmenu select event
container.documentEditor.customContextMenuSelect = (args: CustomContentMenuEventArgs): void =>
{
let item: string = args.id;
let id: string = container.documentEditor.element.id;
switch (item) {
case id + 'searchingoogle':
// To get the selected content as plain text
```

```
let searchContent: string = container.documentEditor.selection.text;
if (!container.documentEditor.selection.isEmpty && /\S/.test(searchContent)) {
window.open('http://google.com/search?q=' + searchContent);
}
break;
}
};
`
```

You can add the following custom options using this API,

- Save or export the selected text as text file.
- Search the selected text in Google or other search engines.
- Show synonyms for the selected word in context menu and replace with selected synonym using the setter method of same API.

Get the selected content as SFDT (rich text)

You can use [sfdt](#) API to get the selected content as plain text from JavaScript Document Editor component.

The following example code illustrates how to get the content of a bookmark and export it as SFDT.

```
`ts
import { DocumentEditorContainer, Toolbar } from '@syncfusion/ej2-documenteditor';
DocumentEditorContainer.Inject(Toolbar);

let container: DocumentEditorContainer = new DocumentEditorContainer({ enableToolbar: true, height:
'590px' });
container.serviceUrl = 'https://services.syncfusion.com/js/production/api/documenteditor/';
container.appendTo('#container');
// To insert text in cursor position
container.documentEditor.editor.insertText('Document editor');
// To select all the content in document
container.documentEditor.selection.selectAll();
// Insert bookmark to selected content
container.documentEditor.editor.insertBookmark('Bookmark1');
//Select the bookmark
container.documentEditor.selection.selectBookmark('Bookmark1');
// To get the selected content as sfdt
let selectedContent: string = container.documentEditor.selection.sfdt;
```

```
// Insert the sfdt content in cursor position using paste API
container.documentEditor.editor.paste(selectedContent);
`ts
```

You can add the following custom options using this API,

- Save or export the selected content as SFDT file.
- Get the content of a bookmark in Word document as SFDT by selecting a bookmark using [selectbookmark](#) API.
- Create template content that can be inserted to multiple documents in cursor position using [paste](#) API.

Set default format in document editor in `##Platform_Name##` Document editor control

You can set the default character format, paragraph format and section format in Document editor.

Set the default character format

You can use [setDefaultCharacterFormat](#) method to set the default character format. For example, Document editor default font size is 11 and you can change it as any valid value.

The following example code illustrates how to change the default font size in Document editor.

```
`ts
import { DocumentEditorContainer, Toolbar } from '@syncfusion/ej2-documenteditor';
let container: DocumentEditorContainer = new DocumentEditorContainer({ height: "590px" });
container.serviceUrl = 'https://services.syncfusion.com/js/production/api/documenteditor/';
DocumentEditorContainer.Inject(Toolbar);
// Default font size set as 20
container.setDefaultCharacterFormat({ fontSize: 20 });
container.appendTo('#container');
`ts
```

Similarly, you can change the required [CharacterFormatProperties](#) default value.

The following example code illustrates how to change other character format default value in Document editor.

```
`ts
import { CharacterFormatProperties, DocumentEditorContainer, Toolbar } from '@syncfusion/ej2-documenteditor';
let container: DocumentEditorContainer = new DocumentEditorContainer({ height: "590px" });
container.serviceUrl = 'https://services.syncfusion.com/js/production/api/documenteditor/';
DocumentEditorContainer.Inject(Toolbar);
// Set default value
let defaultCharacterFormat: CharacterFormatProperties = {
```

```

bold: false,
italic: false,
baselineAlignment: 'Normal',
underline: 'None',
fontColor: "#000000",
fontFamily: 'Algerian',
fontSize: 12
};
container.setDefaultCharacterFormat(defaultCharacterFormat);
container.appendTo('#container');
`

```

Set the default paragraph format

You can use [setDefaultParagraphFormat](#) API to set the default paragraph format. You can change the required [ParagraphFormatProperties](#) default value.

The following example code illustrates how to change the paragraph format(before spacing, line spacing etc.,) default value in Document editor.

```

`ts
import { ParagraphFormatProperties, DocumentEditorContainer, Toolbar } from '@syncfusion/ej2-
documenteditor';

let container: DocumentEditorContainer = new DocumentEditorContainer({ height: "590px" });
container.serviceUrl = 'https://services.syncfusion.com/js/production/api/documenteditor/';
DocumentEditorContainer.Inject(Toolbar);

let defaultParagraphFormat: ParagraphFormatProperties = {
beforeSpacing: 8,
lineSpacing: 1.5,
leftIndent: 24,
textAlignment: "Center"
};

container.setDefaultParagraphFormat(defaultParagraphFormat);
container.appendTo('#container');
`

```

Set the default section format

You can use [setDefaultSectionFormat](#) API to set the default section format. You can change the required [SectionFormatProperties](#) default value.

The following example code illustrates how to change the section format(header and footer distance, page width and height, etc.,) default value in Document editor.

```
`ts
import { SectionFormatProperties, DocumentEditorContainer, Toolbar } from '@syncfusion/ej2-
documenteditor';

let container: DocumentEditorContainer = new DocumentEditorContainer({ height: "590px" });
container.serviceUrl = 'https://services.syncfusion.com/js/production/api/documenteditor/';
DocumentEditorContainer.Inject(Toolbar);

let defaultSectionFormat: SectionFormatProperties = {
  pageWidth: 500,
  pageHeight: 800,
  headerDistance: 56,
  footerDistance: 48,
  leftMargin: 12,
  rightMargin: 12,
  topMargin: 0,
  bottomMargin: 0
};

container.setDefaultSectionFormat(defaultSectionFormat);
container.appendTo('#container');
```

Show hide spinner in ##Platform_Name## Document editor control

Using [spinner](#) component, you can show/hide spinner while opening document in DocumentEditor .

Example code snippet to show/hide spinner

```
`ts
// showSpinner() will make the spinner visible
showSpinner(document.getElementById('container'));

// hideSpinner() method used hide spinner
hideSpinner(document.getElementById('container'));
```

Refer to the following example.

INDEX.JS

```
ej.documenteditor.DocumentEditorContainer.Inject(ej.documenteditor.Toolbar);
var container = new ej.documenteditor.DocumentEditorContainer({
  enableToolbar: true, height:"400"
```

```

});
ej.popups.createSpinner({
    // Specify the target for the spinner to show
    target: document.getElementById('container')
});
document.getElementById('import').addEventListener('click',function(){
    // load your default document here
    var data=
    '{"sections":[{"sectionFormat":{"pageWidth":612,"pageHeight":792,"leftMargin":72,"rightMargin":72,"topMargin":72,"bottomMargin":72,"differentFirstPage":false,"differentOddAndEvenPages":false,"headerDistance":36,"footerDistance":36,"bidi":false},"blocks":[{"paragraphFormat":{"afterSpacing":30,"styleName":"Heading 1","listFormat":{},"characterFormat":{},"inlines":[{"characterFormat":{},"text":"Adventure Works Cycles"}]}]}],"headersFooters":{"header":{"blocks":[{"paragraphFormat":{"listFormat":{},"characterFormat":{},"inlines":[]}}],"footer":{"blocks":[{"paragraphFormat":{"listFormat":{},"characterFormat":{},"inlines":[]}}]}],"characterFormat":{"bold":false,"italic":false,"fontSize":11,"fontFamily":"Calibri","underline":"None","strikethrough":"None","baselineAlignment":"Normal","highlightColor":"NoColor","fontColor":"empty","fontSizeBidi":11,"fontFamilyBidi":"Calibri","allCaps":false},"paragraphFormat":{"leftIndent":0,"rightIndent":0,"firstLineIndent":0,"textAlignment":"Left","beforeSpacing":0,"afterSpacing":0,"lineSpacing":1.0791666507720947,"lineSpacingType":"Multiple","listFormat":{},"bidi":false},"defaultTabWidth":36,"trackChanges":false,"enforcement":false,"hashValue":"","saltValue":"","formatting":false,"protectionType":"NoProtection","dontUseHTMLParagraphAutoSpacing":false,"formFieldShading":true,"styles":[{"name":"Normal","type":"Paragraph","paragraphFormat":{"lineSpacing":1.149999976158142,"lineSpacingType":"Multiple","listFormat":{},"characterFormat":{"fontFamily":"Calibri"},"next":"Normal"},{"name":"Default Paragraph Font","type":"Character","characterFormat":{"fontSize":16,"fontFamily":"Calibri Light","fontColor":"#2F5496"},"basedOn":"Default Paragraph Font"},{"name":"Heading 1","type":"Paragraph","paragraphFormat":{"beforeSpacing":12,"afterSpacing":0,"outlineLevel":"Level1","listFormat":{},"characterFormat":{"fontSize":16,"fontFamily":"Calibri Light","fontColor":"#2F5496"},"basedOn":"Normal","link":"Heading 1 Char","next":"Normal"},{"name":"Heading 2 Char","type":"Character","characterFormat":{"fontSize":13,"fontFamily":"Calibri Light","fontColor":"#2F5496"},"basedOn":"Default Paragraph Font"},{"name":"Heading 2","type":"Paragraph","paragraphFormat":{"beforeSpacing":2,"afterSpacing":6,"outlineLevel":"Level2","listFormat":{},"characterFormat":{"fontSize":13,"fontFamily":"Calibri Light","fontColor":"#2F5496"},"basedOn":"Normal","link":"Heading 2 Char","next":"Normal"},{"name":"Heading 3","type":"Paragraph","paragraphFormat":{"leftIndent":0,"rightIndent":0,"firstLineIndent":0,"textAlignment":"Left","beforeSpacing":2,"afterSpacing":0,"lineSpacing":1.0791666507720947,"lineSpacingType":"Multiple","outlineLevel":"Level3","listFormat":{},"characterFormat":{"fontSize":12,"fontFamily":"Calibri Light","fontColor":"#1F3763"},"basedOn":"Normal","link":"Heading 3 Char","next":"Normal"},{"name":"Heading 3 Char","type":"Character","characterFormat":{"fontSize":12,"fontFamily":"Calibri Light","fontColor":"#1F3763"},"basedOn":"Default Paragraph Font"},{"name":"Heading 4","type":"Paragraph","paragraphFormat":{"leftIndent":0,"rightIndent":0,"fir

```

```

stLineIndent":0,"textAlignment":"Left","beforeSpacing":2,"afterSpacing":0,"l
ineSpacing":1.0791666507720947,"lineSpacingType":"Multiple","outlineLevel":"
Level4","listFormat":{},"characterFormat":{"italic":true,"fontFamily":"Cali
bri Light","fontColor":"#2F5496"},"basedOn":"Normal","link":"Heading 4
Char","next":"Normal"}, {"name":"Heading 4
Char","type":"Character","characterFormat":{"italic":true,"fontFamily":"Cali
bri Light","fontColor":"#2F5496"},"basedOn":"Default Paragraph
Font"}, {"name":"Heading
5","type":"Paragraph","paragraphFormat":{"leftIndent":0,"rightIndent":0,"fir
stLineIndent":0,"textAlignment":"Left","beforeSpacing":2,"afterSpacing":0,"l
ineSpacing":1.0791666507720947,"lineSpacingType":"Multiple","outlineLevel":"
Level5","listFormat":{},"characterFormat":{"fontFamily":"Calibri
Light","fontColor":"#2F5496"},"basedOn":"Normal","link":"Heading 5
Char","next":"Normal"}, {"name":"Heading 5
Char","type":"Character","characterFormat":{"fontFamily":"Calibri
Light","fontColor":"#2F5496"},"basedOn":"Default Paragraph
Font"}, {"name":"Heading
6","type":"Paragraph","paragraphFormat":{"leftIndent":0,"rightIndent":0,"fir
stLineIndent":0,"textAlignment":"Left","beforeSpacing":2,"afterSpacing":0,"l
ineSpacing":1.0791666507720947,"lineSpacingType":"Multiple","outlineLevel":"
Level6","listFormat":{},"characterFormat":{"fontFamily":"Calibri
Light","fontColor":"#1F3763"},"basedOn":"Normal","link":"Heading 6
Char","next":"Normal"}, {"name":"Heading 6
Char","type":"Character","characterFormat":{"fontFamily":"Calibri
Light","fontColor":"#1F3763"},"basedOn":"Default Paragraph
Font"}], "lists": [], "abstractLists": [], "comments": [], "revisions": [], "customXm
l": []}';
ej.popups.showSpinner(document.getElementById('DocumentEditor'));
// Open the default document
container.documentEditor.open(data);
setInterval(function() {
// hideSpinner() method used hide spinner
ej.popups.hideSpinner(document.getElementById('DocumentEditor'));
}, 5000);
});
container.appendTo('#DocumentEditor');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
documenteditor/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/fabric.css" rel="stylesheet">

```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="import">Load Document</button>
        <div id="DocumentEditor" style="height:420px">

            </div>
        </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Note: In above example, we have used setInterval to hide spinner, just for demo purpose.

Resize document editor in `##Platform_Name##` Document editor control

In this article, we are going to see how to change height and width of DocumentEditor.

Change height of Document Editor

DocumentEditorContainer initially render with default height. You can change height of documenteditor using [height](#) property, the value which is in pixel.

The following example code illustrates how to change height of Document Editor.

```
`ts
```

```

let container: DocumentEditorContainer = new DocumentEditorContainer({
enableToolbar: true, height: "590px"
});
container.appendTo('#DocumentEditor');
`

```

Similarly, you can use [height](#) property for DocumentEditor also.

Change width of Document Editor

DocumentEditorContainer initially render with default width. You can change width of documenteditor using [width](#) property, the value which is in percent.

The following example code illustrates how to change width of Document Editor.

```
`ts
let container: DocumentEditorContainer = new DocumentEditorContainer({
enableToolbar: true, width: "100%"
});
container.appendTo('#DocumentEditor');
```

Similarly, you can use [width](#) property for DocumentEditor also.

Resize Document Editor

Using [resize](#) method, you change height and width of Document editor.

The following example code illustrates how to fit Document Editor to browser window size.

```
`ts
import {
DocumentEditorContainer,
Toolbar,
} from '@syncfusion/ej2-documenteditor';

let container: DocumentEditorContainer = new DocumentEditorContainer({ enableToolbar: true, height:
'590px' });

DocumentEditorContainer.Inject(Toolbar);

container.serviceUrl =
'https://services.syncfusion.com/js/production/api/documenteditor/';

container.created = (): void => {
setInterval(() => {
updateDocumentEditorSize();
}, 100);

//Adds event listener for browser window resize event.
window.addEventListener('resize', onWindowResize);
};

container.appendTo('#container');

function onWindowResize() {

//Resizes the document editor component to fit full browser window automatically whenever the
browser resized.
```

```

updateDocumentEditorSize();
}
function updateDocumentEditorSize() {
//Resizes the document editor component to fit full browser window.
var windowWidth = window.innerWidth;
var windowHeight = window.innerHeight;
container.resize(windowWidth, windowHeight);
}
`

```

Export document as pdf in ##Platform_Name## Document editor control

In this article, we are going to see how to export the document as Pdf format. You can export the document as Pdf in following ways:

Export the document as pdf in client-side

Use [pdf export component](#) in application level to export the document as pdf using [exportasimage](#) API. Here, all pages will be converted to image and inserted as pdf pages(works like print as PDF).

Note:

- You can install the pdf export packages from this [link](#).
- There is one limitation we can't search the text because we are exporting the pdf as image.

The following example code illustrates how to export the document as pdf in client-side.

```

`ts
import {
DocumentEditorContainer,
ImageFormat,
Toolbar,
} from '@syncfusion/ej2-documenteditor';
import {
PdfBitmap,
PdfDocument,
PdfPageOrientation,
PdfPageSettings,
PdfSection,
SizeF,
} from '@syncfusion/ej2-pdf-export';
let container: DocumentEditorContainer = new DocumentEditorContainer({

```

```

enableToolbar: true,
height: '590px',
});
DocumentEditorContainer.Inject(Toolbar);
container.serviceUrl =
'https://services.syncfusion.com/js/production/api/documenteditor/';
container.appendTo('#container');
document.getElementById('export').addEventListener('click', function () {
let obj = this;
let pdfdocument: PdfDocument = new PdfDocument();
let count: number = container.documentEditor.pageCount;
container.documentEditor.documentEditorSettings.printDevicePixelRatio = 2;
let loadedPage = 0;
for (let i = 1; i <= count; i++) {
setTimeout(() => {
let format: ImageFormat = 'image/jpeg' as ImageFormat;
// Getting pages as image
let image = container.documentEditor.exportAsImage(i, format);
image.onload = function () {
let imageHeight = parseInt(
image.style.height.toString().replace('px', ''
);
let imageWidth = parseInt(
image.style.width.toString().replace('px', ''
);
let section: PdfSection = pdfdocument.sections.add() as PdfSection;
let settings: PdfPageSettings = new PdfPageSettings(0);
if (imageWidth > imageHeight) {
settings.orientation = PdfPageOrientation.Landscape;
}
settings.size = new SizeF(imageWidth, imageHeight);
(section as PdfSection).setPageSettings(settings);
let page = section.pages.add();

```

```

let graphics = page.graphics;
let imageStr = image.src.replace('data:image/jpeg;base64,', '');
let pdfImage = new PdfBitmap(imageStr);
graphics.drawImage(pdfImage, 0, 0, imageWidth, imageHeight);
loadedPage++;
if (loadedPage == count) {
// Exporting the document as pdf
pdfdocument.save(
(container.documentEditor.documentName === ''
? 'sample'
: container.documentEditor.documentName) + '.pdf'
);
}
};
}, 500);
}
});
`

```

Export document as pdf in server-side using Syncfusion DocIO

With the help of [Syncfusion DocIO](#), you can export the document as Pdf in server-side. Here, you can search the text.

The following way illustrates how to convert the document as Pdf:

- Using [serialize](#) API, convert the document as Sfdt and send it to server-side.

The following example code illustrates how to convert the document to sfdt and pass it to server-side.

```

`ts
import { DocumentEditorContainer, Toolbar } from '@syncfusion/ej2-documenteditor';
let container: DocumentEditorContainer = new DocumentEditorContainer({
enableToolbar: true,
height: '590px',
});
DocumentEditorContainer.Inject(Toolbar);
container.serviceUrl =
'https://services.syncfusion.com/js/production/api/documenteditor/';

```

```

container.appendTo('#container');
document.getElementById('export').addEventListener('click', function () {
let http: XMLHttpRequest = new XMLHttpRequest();
// Replace your running web service Url here
http.open('POST', 'http://localhost:62869/api/documenteditor/ExportPdf');
http.setRequestHeader('Content-Type', 'application/json;charset=UTF-8');
http.responseType = 'json';
//Serialize document content as SFDt.
let sfdt: any = { content: container.documentEditor.serialize() };
//Send the sfdt content to server side.
http.send(JSON.stringify(sfdt));
});
`

```

- Using Save API in server-side, you can convert the sfdt to stream.
- Finally, convert the stream to Pdf using [Syncfusion.DocIO.Renderer.Net.Core](#) library.

The following example code illustrates how to process the sfdt in server-side.

```

`c#
[AcceptVerbs("Post")]
[HttpPost]
[EnableCors("AllowAllOrigins")]
[Route("ExportPdf")]
public void ExportPdf([FromBody]SaveParameter data)
{
// Converts the sfdt to stream
Stream document = WordDocument.Save(data.content, FormatType.Docx);
Syncfusion.DocIO.DLS.WordDocument doc = new Syncfusion.DocIO.DLS.WordDocument(document,
Syncfusion.DocIO.FormatType.Docx);
//Instantiation of DocIO.Renderer for Word to PDF conversion
DocIO.Renderer render = new DocIO.Renderer();
//Converts Word document into PDF document
PdfDocument pdfDocument = render.ConvertToPDF(doc);
// Saves the document to server machine file system, you can customize here to save into databases or
file servers based on requirement.

```

```
FileStream fileStream = new FileStream("sample.pdf", FileMode.OpenOrCreate, FileAccess.ReadWrite);  
//Saves the PDF file  
pdfDocument.Save(fileStream);  
pdfDocument.Close();  
fileStream.Close();  
document.Close();  
}  
`
```

Get the complete working sample in this [link](#).

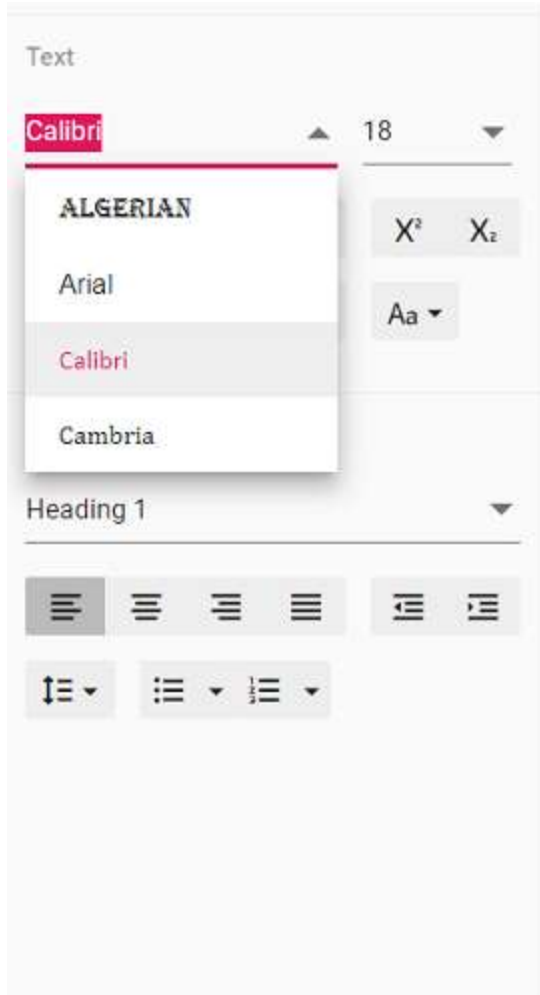
Customize font family drop down in `##Platform_Name## Document editor control`
Document editor provides an options to customize the font family drop down list values using [fontfamilies](#) in Document editor settings. Fonts which are added in fontFamilies of [documentEditorSettings](#) will be displayed on font drop down list of text properties pane and font dialog.

Similarly, you can use [documentEditorSettings](#) property for DocumentEditor also.

The following example code illustrates how to change the font families in Document editor container.

```
`ts  
let container: DocumentEditorContainer = new DocumentEditorContainer({ enableToolbar: true,height:  
'590px',  
// Add required font families to list it in font drop down  
documentEditorSettings: {  
fontFamilies: ['Algerian', 'Arial', 'Calibri', 'Windings'],  
}  
});  
DocumentEditorContainer.Inject(Toolbar);  
container.serviceUrl = 'https://services.syncfusion.com/js/production/api/documenteditor/';  
container.appendTo('#container');  
`
```

Output will be like below:



Auto save document in document editor in `##Platform_Name##` Document editor control

In this article, we are going to see how to autosave the document in AWS S3. You can automatically save the edited content in regular intervals of time. It helps reduce the risk of data loss by saving an open document automatically at customized intervals.

The following example illustrates how to auto save the document in AWS S3.

- In the client-side, using content change event, we can automatically save the edited content in regular intervals of time. Based on `contentChanged` boolean, the document send as Docx format to server-side using [saveAsBlob](#) method.

```
`ts
import {
  DocumentEditorContainer,
  Toolbar,
} from '@syncfusion/ej2-documenteditor';
```



```

let container: DocumentEditorContainer = new DocumentEditorContainer({ enableToolbar: true, height:
'590px' });
let contentChanged: boolean = false;
DocumentEditorContainer.Inject(Toolbar);
container.serviceUrl = 'https://services.syncfusion.com/js/production/api/documenteditor/';
container.created = function () {
  setInterval(() => {
    if (contentChanged) {
      //You can save the document as below
      container.documentEditor.saveAsBlob('Docx').then((blob: Blob) => {
        console.log('Saved sucessfully');
        let exportedDocument: Blob = blob;
        //Now, save the document where ever you want.
        let formData: FormData = new FormData();
        formData.append('fileName', 'sample.docx');
        formData.append('data', exportedDocument);
        / tslint:disable /
        var req = new XMLHttpRequest();
        // Replace your running Url here
        req.open(
          'POST',
          'http://localhost:62869/api/documenteditor/SaveToS3',
          true
        );
        req.onreadystatechange = () => {
          if (req.readyState === 4) {
            if (req.status === 200 || req.status === 304) {
              console.log('Saved sucessfully');
            }
          }
        };
        req.send(formData);
      });
    }
  });
}

```

```

contentChanged = false;
}
}, 1000);
};
container.appendTo('#container');
container.contentChange = (): void => {
contentChanged = true;
};
`c#

```

- In server-side, configure the access key and secret key in `web.config` file and register profile in `startup.cs`.

In `web.config`, add key like below format:

```

`c#
<appSettings>
<add key="AWSProfileName" value="sync_development" />
<add key="AWSAccessKey" value="" />
<add key="AWSSecretKey" value="" />
</appSettings>
`c#

```

In `startup.cs`, register profile in below format:

```

`c#
Amazon.Util.ProfileManager.RegisterProfile("sync_development", "", "");
`c#

```

- In server-side, Receives the stream content from client-side and process it to save the document in aws s3. Add Web API in controller file like below to save the document in aws s3.

```

`c#
[AcceptVerbs("Post")]
[HttpPost]
[EnableCors("AllowAllOrigins")]
[Route("SaveToS3")]
public string SaveToS3()
{

```

```

IFormFile file = HttpContext.Request.Form.Files[0];
Stream stream = new MemoryStream();
file.CopyTo(stream);
UploadFileStreamToS3(stream, "documenteditor", "", "GettingStarted.docx");
stream.Close();
return "Sucess";
}

public bool UploadFileStreamToS3(System.IO.Stream localFilePath, string bucketName, string
subDirectoryInBucket, string fileNameInS3)
{
    AWSCredentials credentials = new StoredProfileAWSCredentials("sync_development");
    IAmazonS3 client = new AmazonS3Client(credentials, Amazon.RegionEndpoint.USEast1);
    TransferUtility utility = new TransferUtility(client);
    TransferUtilityUploadRequest request = new TransferUtilityUploadRequest();
    if (subDirectoryInBucket == "" || subDirectoryInBucket == null)
    {
        request.BucketName = bucketName; //no subdirectory just bucket name
    }
    else
    {
        // subdirectory and bucket name
        request.BucketName = bucketName + @"/" + subDirectoryInBucket;
    }
    request.Key = fileNameInS3; //file name up in S3
    request.InputStream = localFilePath;
    utility.Upload(request); //commencing the transfer
    return true; //indicate that the file was sent
}

```

Get the complete working sample in this [link](#).

Retrieve the bookmark content as text in `##Platform_Name##` Document editor control

You can get the bookmark or whole document content from the JavaScript Document Editor component as plain text and SFDT (rich text).

Get the bookmark content as plain text

You can [selectBookmark](#) API to navigate to the bookmark and use [text](#) API to get the bookmark content as plain text from JavaScript Document Editor component.

The following example code illustrates how to get the bookmark content as plain text.

`ts

```
import { DocumentEditorContainer, Toolbar } from '@syncfusion/ej2-documenteditor';
DocumentEditorContainer.Inject(Toolbar);

let container: DocumentEditorContainer = new DocumentEditorContainer({ enableToolbar: true, height:
'590px' });

container.serviceUrl = 'https://services.syncfusion.com/js/production/api/documenteditor/';
container.appendTo('#container');

// To insert text in cursor position
container.documentEditor.editor.insertText('Document editor');

// To select all the content in document
container.documentEditor.selection.selectAll();

// Insert bookmark to selected content
container.documentEditor.editor.insertBookmark('Bookmark1');

// Provide your bookmark name to navigate to specific bookmark
container.documentEditor.selection.selectBookmark('Bookmark1');

// To get the selected content as text
let selectedContent: string = container.documentEditor.selection.text;
`
```

To get the bookmark content as SFDT (rich text), please check this [link](#)

[Get the whole document content as text](#)

You can use [text](#) API to get the whole document content as plain text from JavaScript Document Editor component.

The following example code illustrates how to get the whole document content as plain text.

`ts

```
import { DocumentEditorContainer, Toolbar } from '@syncfusion/ej2-documenteditor';
DocumentEditorContainer.Inject(Toolbar);

let container: DocumentEditorContainer = new DocumentEditorContainer({ enableToolbar: true, height:
'590px' });

container.serviceUrl = 'https://services.syncfusion.com/js/production/api/documenteditor/';
container.appendTo('#container');

// To insert text in cursor position
container.documentEditor.editor.insertText('Document editor');

// To select all the content in document
```

```
container.documentEditor.selection.selectAll();
```

```
// To get the content as text
```

```
let selectedContent: string = container.documentEditor.selection.text;
```

```
,
```

Get the whole document content as SFDT(rich text)

You can use [serialize](#) API to get the whole document content as SFDT string from JavaScript Document Editor component.

The following example code illustrates how to get the whole document content as SFDT.

```
`ts
```

```
import { DocumentEditorContainer, Toolbar } from '@syncfusion/ej2-documenteditor';
```

```
DocumentEditorContainer.Inject(Toolbar);
```

```
let container: DocumentEditorContainer = new DocumentEditorContainer({ enableToolbar: true, height: '590px' });
```

```
container.serviceUrl = 'https://services.syncfusion.com/js/production/api/documenteditor/';
```

```
container.appendTo('#container');
```

```
// To insert text in cursor position
```

```
container.documentEditor.editor.insertText('Document editor');
```

```
// To get the content as SFDT
```

```
let selectedContent: string = container.documentEditor.serialize();
```

```
,
```

Get the header content as text

You can use [goToHeader](#) API to navigate the selection to the header and then use [text](#) API to get the content as plain text.

The following example code illustrates how to get the header content as plain text.

```
`ts
```

```
import { DocumentEditorContainer, Toolbar } from '@syncfusion/ej2-documenteditor';
```

```
DocumentEditorContainer.Inject(Toolbar);
```

```
let container: DocumentEditorContainer = new DocumentEditorContainer({ enableToolbar: true, height: '590px' });
```

```
container.serviceUrl = 'https://services.syncfusion.com/js/production/api/documenteditor/';
```

```
container.appendTo('#container');
```

```
// To navigate the selection to header
```

```
container.documentEditor.selection.goToHeader();
```

```
// To insert text in cursor position
```

```
container.documentEditor.editor.insertText('Document editor');
```

```
// To select all the content in document
container.documentEditor.selection.selectAll();

// To get the selected content as text
let selectedContent: string = container.documentEditor.selection.text;
`
```

Similarly, you can use [goToFooter](#) API to navigate the selection to the footer and then use [text](#) API to get the content as plain text.

Get current word in ##Platform_Name## Document editor control

You can get the current word or paragraph content from the JavaScript Document Editor component as plain text and SFDT (rich text).

Select and get the word in current cursor position

You can use [selectCurrentWord](#) API in selection module to select the current word at cursor position and use [text](#) API to get the selected content as plain text from JavaScript Document Editor component.

The following example code illustrates how to select and get the current word as plain text.

```
`ts
import { DocumentEditorContainer, Toolbar } from '@syncfusion/ej2-documenteditor';

DocumentEditorContainer.Inject(Toolbar);

let container: DocumentEditorContainer = new DocumentEditorContainer({ enableToolbar: true, height:
'590px' });

container.serviceUrl = 'https://services.syncfusion.com/js/production/api/documenteditor/';
container.appendTo('#container');

// To insert text in cursor position
container.documentEditor.editor.insertText('Document editor');

// To select the current word in document
container.documentEditor.selection.selectCurrentWord();

// To get the selected content as text
let selectedContent: string = container.documentEditor.selection.text;
`
```

To get the bookmark content as SFDT (rich text), please check this [link](#)

Select and get the paragraph in current cursor position

You can use [selectParagraph](#) API in selection module to select the current paragraph at cursor position and use [text](#) API or [sfdt](#) API to get the selected content as plain text or SFDT from JavaScript Document Editor component.

The following example code illustrates how to select and get the current paragraph as SFDT.

```
`ts
import { DocumentEditorContainer, Toolbar } from '@syncfusion/ej2-documenteditor';
```

```

DocumentEditorContainer.Inject(Toolbar);

let container: DocumentEditorContainer = new DocumentEditorContainer({ enableToolbar: true, height:
'590px' });

container.serviceUrl = 'https://services.syncfusion.com/js/production/api/documenteditor/';
container.appendTo('#container');

// To insert text in cursor position
container.documentEditor.editor.insertText('Document editor');

// To select the current paragraph in document
container.documentEditor.selection.selectParagraph();

// To get the selected content as SFDT
let selectedContent: string = container.documentEditor.selection.sfdt;
`

```

Insert page number and navigate to page in ##Platform_Name## Document editor control

You can insert page number and navigate to specific page in JavaScript Document Editor component by following ways.

Insert page number

You can use [insertPageNumber](#) API in editor module to insert the page number in current cursor position. By default, Page number will insert in Arabic number style. You can change it, by providing the number style in parameter.

Note: Currently, Documenteditor have options to insert page number at current cursor position.

The following example code illustrates how to insert page number in header.

```

`ts
import { DocumentEditorContainer, Toolbar } from '@syncfusion/ej2-documenteditor';

DocumentEditorContainer.Inject(Toolbar);

let container: DocumentEditorContainer = new DocumentEditorContainer({ enableToolbar: true, height:
'590px' });

container.serviceUrl = 'https://services.syncfusion.com/js/production/api/documenteditor/';
container.appendTo('#container');

// To insert text in cursor position
container.documentEditor.editor.insertText('Document editor');

// To move the selection to header
container.documentEditor.selection.goToHeader();

// Insert page number in the current cursor position
container.documentEditor.editor.insertPageNumber();
`

```

Also, you use [insertField](#) API in Editor module to insert the Page number in current position

```
`ts
//Current page number
container.documentEditor.editor.insertField('PAGE * MERGEFORMAT', '1');
`
```

Get page count

You can use [pageCount](#) API to gets the total number of pages in Document.

The following example code illustrates how to get the number of pages in Document.

```
`ts
import { DocumentEditorContainer, Toolbar } from '@syncfusion/ej2-documenteditor';
DocumentEditorContainer.Inject(Toolbar);

let container: DocumentEditorContainer = new DocumentEditorContainer({ enableToolbar: true, height:
'590px' });

container.serviceUrl = 'https://services.syncfusion.com/js/production/api/documenteditor/';
container.appendTo('#container');

// To insert text in cursor position
container.documentEditor.editor.insertText('Document editor');

// To get the total number of pages
let pageCount : number=container.documentEditor.pageCount;
`
```

Navigate to specific page

You can use [goToPage](#) API in Selection module to move selection to the start of the specified page number.

The following example code illustrates how to move selection to specific page.

```
`ts
import { DocumentEditorContainer, Toolbar } from '@syncfusion/ej2-documenteditor';
DocumentEditorContainer.Inject(Toolbar);

let container: DocumentEditorContainer = new DocumentEditorContainer({ enableToolbar: true, height:
'590px' });

container.serviceUrl = 'https://services.syncfusion.com/js/production/api/documenteditor/';
container.appendTo('#container');

// To move selection to page number 2
container.documentEditor.selection.goToPage(2);
`
```


Move selection to specific position in `##Platform_Name##` Document editor control
Using [select](#) API in selection module, You can set cursor position to anywhere in the document.

Selects content based on start and end hierarchical index

Hierarchical index will be in below format.

`sectionIndex;blockIndex;offset`

The following code snippet illustrate how to select using hierarchical index.

```
`ts
// Selection will occur between provided start and end offset
documentEdContainerIns.documentEditor.editor.insertText("Welcome");
// The below code will select the letters "We" from inserted text "Welcome"
documentEdContainerIns.documentEditor.selection.select("0;0;0", "0;0;2");
`
```

The following table illustrates about Hierarchical index in detail.

Element	Hierarchical Format	Explanation
Move selection to Paragraph	sectionIndex;blockIndex;offset	 Ex: 0;0;0 It moves the cursor to the start of paragraph.
Move selection to Table	sectionIndex;tableIndex;rowIndex;cellIndex;blockIndex;offset	 Ex: 0;0;0;1;0 It moves the cursor to the second paragraph which is inside first row and cell of table.
Move selection to header	pageIndex;H;sectionIndex;blockIndex;offset	 Ex: 1;H;0;0;0 It moves cursor to the header in second page.
Move selection to Footer	pageIndex;F;sectionIndex;blockIndex;offset	 Ex: 1;F;0;0;0 It moves cursor to the footer in second page.

Get the selection start and end hierarchical index

Using [startOffset](#), you can get start hierarchical index which denotes the start index of current selection. Similarly, using [endOffset](#), you can get end hierarchical index which denotes the end index of current selection.

The following code snippet illustrate how to get the selection start and end offset on selection changes in document.

```
`ts
import { DocumentEditorContainer, Toolbar } from '@syncfusion/ej2-documenteditor';
let hostUrl: string = 'https://ej2services.syncfusion.com/production/web-services/';
let container: DocumentEditorContainer = new DocumentEditorContainer({ enableToolbar: true, height: '590px' });
DocumentEditorContainer.Inject(Toolbar);
container.serviceUrl = hostUrl + 'api/documenteditor/';
```

```
container.appendTo('#container');
// Event gets triggered on selection change in document
container.selectionChange = (): void => {
//Get the start index of current selection
let startOffset:string = container.documentEditor.selection.startOffset;
//Get the end index of current selection
let endOffset:string = container.documentEditor.selection.endOffset;
};
`
```

Document editor have [selectionChange](#) event which is triggered whenever the selection changes in Document.

Selects the content based on left and top position

Here, you can specify the [selection settings](#) to select the content based on left and top position.

x denotes the left position and y denotes the top position and extend denotes whether to extend or update selection.

Please check below code sample for reference.

```
`ts
Container.documentEditor.selection.select({ x: 188.4814208984375 , y: 662.00005, extend: true });
`
```

Disable header and footer edit in document editor in `##Platform_Name##` Document editor control

Disable header and footer edit in DocumentEditorContainer instance

You can use [restrictEditing](#) property to disable header and footer editing based on selection context type.

RestrictEditing allows you to restrict the document modification and makes the Document read only mode. So, by using this property, and if selection inside header or footer, you can set this property as true.

The following example code illustrates how to header and footer edit in `DocumentEditorContainer` instance.

```
`ts
import { DocumentEditorContainer, Toolbar } from '@syncfusion/ej2-documenteditor';
let hostUrl: string = 'https://ej2services.syncfusion.com/production/web-services/';
let container: DocumentEditorContainer = new DocumentEditorContainer({ enableToolbar: true, height: '590px' });
DocumentEditorContainer.Inject(Toolbar);
container.serviceUrl = hostUrl + 'api/documenteditor/';
`
```

```

container.appendTo('#container');
container.selectionChange = (): void => {
// Check whether selection is in header
if (container.documentEditor.selection.contextType.indexOf('Header') > -1 ||
// Check whether selection is in Footer
container.documentEditor.selection.contextType.indexOf('Footer') > -1) {
// Change the document to read only mode
container.restrictEditing = true;
} else {
// Change the document to editable mode
container.restrictEditing = false;
}
};
`

```

Otherwise, you can disable clicking inside Header or Footer by using [closeHeaderFooter](#) API in selection module.

The following example code illustrates how to close header and footer when selection is inside header or footer in `DocumentEditorContainer` instance.

```

`ts
import { DocumentEditorContainer, Toolbar } from '@syncfusion/ej2-documenteditor';
let hostUrl: string = 'https://ej2services.syncfusion.com/production/web-services/';
let container: DocumentEditorContainer = new DocumentEditorContainer({ enableToolbar: true, height:
'590px' });
DocumentEditorContainer.Inject(Toolbar);
container.serviceUrl = hostUrl + 'api/documenteditor/';
container.appendTo('#container');
container.selectionChange = (): void => {
// Check whether selection is in header
if (container.documentEditor.selection.contextType.indexOf('Header') > -1 ||
// Check whether selection is in Footer
container.documentEditor.selection.contextType.indexOf('Footer') > -1) {
// Close header Footer
container.documentEditor.selection.closeHeaderFooter();
}
}
`

```

```
};  
`
```

Disable header and footer edit in DocumentEditor instance

Like `restrictEditing`, you can use [isReadOnly](#) property in Document editor to disable header and footer edit.

The following example code illustrates how to header and footer edit in `DocumentEditor` instance.

```
`ts  
import { DocumentEditor } from '@syncfusion/ej2-documenteditor';  
let hostUrl: string = 'https://services.syncfusion.com/js/production/';  
let documentEditor: DocumentEditor = new DocumentEditor({ isReadOnly: false, height: '590px' });  
documentEditor.enableAllModules();  
documentEditor.serviceUrl = hostUrl + 'api/documenteditor/';  
documentEditor.appendTo('#documentEditor');  
documentEditor.selectionChange = (): void => {  
    // Check whether selection is in header  
    if (documentEditor.selection.contextType.indexOf('Header') > -1 ||  
    // Check whether selection is in Footer  
    documentEditor.selection.contextType.indexOf('Footer') > -1) {  
        // Change the document to read only mode  
        documentEditor.isReadOnly = true;  
    } else {  
        // Change the document to editable mode  
        documentEditor.isReadOnly = false;  
    }  
};  
`
```

Insert text in current position in ##Platform_Name## Document editor control

You can insert the text, paragraph and rich-text content in JavaScript Document Editor component.

Insert text in current cursor position

You can use [insertText](#) API in editor module to insert the text in current cursor position.

The following example illustrates how to add the text in current selection.

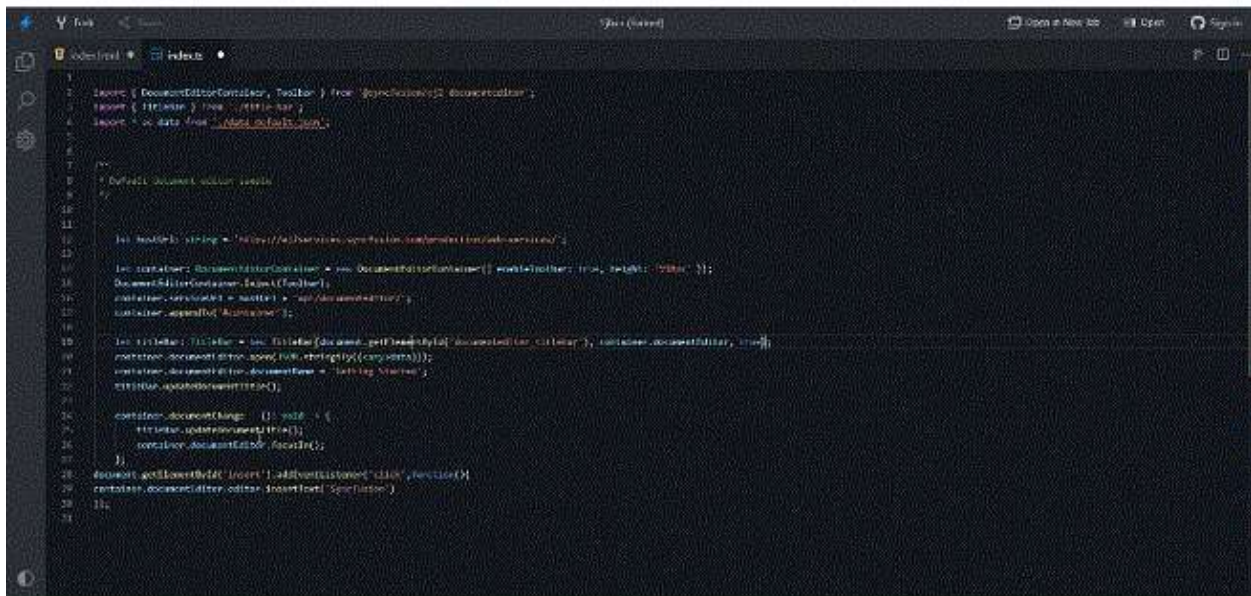
```
`ts  
let hostUrl: string = 'https://ej2services.syncfusion.com/production/web-services/';  
let container: DocumentEditorContainer = new DocumentEditorContainer({ enableToolbar: true, height:  
'590px' });
```

```

DocumentEditorContainer.Inject(Toolbar);
container.serviceUrl = hostUrl + 'api/documenteditor/';
container.appendTo('#container');
document.getElementById('insert').addEventListener('click',function(){
// It will insert the provided text in current selection
container.documentEditor.editor.insertText('SynCFusion');
});

```

Please check below gif which illustrates how to insert text in current cursor position on button click:



Insert paragraph in current cursor position

To insert new paragraph at current selection, you can use [insertText](#) API with parameter as `\r\n` or `\n`.

The following example code illustrates how to add the new paragraph in current selection.

```

`ts
// It will add the new paragraph in current selection
container.documentEditor.editor.insertText('\n');

```

Insert the rich-text content

To insert the HTML content, you have to convert the HTML content to SFDT Format using [web service](#). Then use [paste](#) API to insert the sfdt at current cursor position.

Note: Html string should be wellformatted html. [DocIO](#) support only wellformatted XHTML.

The following example illustrates how to insert the HTML content at current cursor position.

- Send the HTML content to server side for SFDT conversion. Refer to the following example to send the HTML content to server side and inserting it in current cursor position.

```
`ts
import {
  DocumentEditorContainer,
  Toolbar,
} from '@syncfusion/ej2-documenteditor';

let hostUrl: string =
'https://ej2services.syncfusion.com/production/web-services/';

let container: DocumentEditorContainer = new DocumentEditorContainer({
  enableToolbar: true,
  height: '590px',
});

DocumentEditorContainer.Inject(Toolbar);

container.serviceUrl = hostUrl + 'api/documenteditor/';

container.appendTo('#container');

let htmltags: string =
"<?xml version='1.0' encoding='utf - 8'?><!DOCTYPE html PUBLIC '-//W3C//DTD XHTML 1.0 Strict//EN'http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd><html xmlns='http://www.w3.org/1999/xhtml' xml:lang='en' lang='en'><body><h1>The img element</h1><img src='https://www.w3schools.com/images/lamp.jpg' alt='Lamp Image' width='500' height='600'/></body></html>";

document.getElementById('export').addEventListener('click', () => {
  let http: XMLHttpRequest = new XMLHttpRequest();
  http.open('POST', 'http://localhost:5000/api/documenteditor/LoadString');
  http.setRequestHeader('Content-Type', 'application/json;charset=UTF-8');
  http.responseType = 'json';
  http.onreadystatechange = function () {
    if (http.readyState === 4) {
      if (http.status === 200 || http.status === 304) {
        // Insert the sfdt content in cursor position using paste API
        container.documentEditor.editor.paste(http.response);
      }
    }
  };
});
```

```

    } else {
    alert('failed;');
    }
    }
};

let htmlContent: any = { content: htmltags };
http.send(JSON.stringify(htmlContent));
});
`

```

- Please refer the following code example for server-side web implementation for HTML conversion using DocumentEditor.

```

`c#
//API controller for the conversion.
[HttpPost]
public string LoadString([FromBody]InputParameter data)
{
// You can also load HTML file/string from server side.
Syncfusion.EJ2.DocumentEditor.WordDocument document =
Syncfusion.EJ2.DocumentEditor.WordDocument.LoadString(data.content, FormatType.Html); // Convert
the HTML to SFDT format.

string json = Newtonsoft.Json.JsonConvert.SerializeObject(document);
document.Dispose();
return json;
}

public class InputParameter
{
public string content {get; set; }
}
`

```

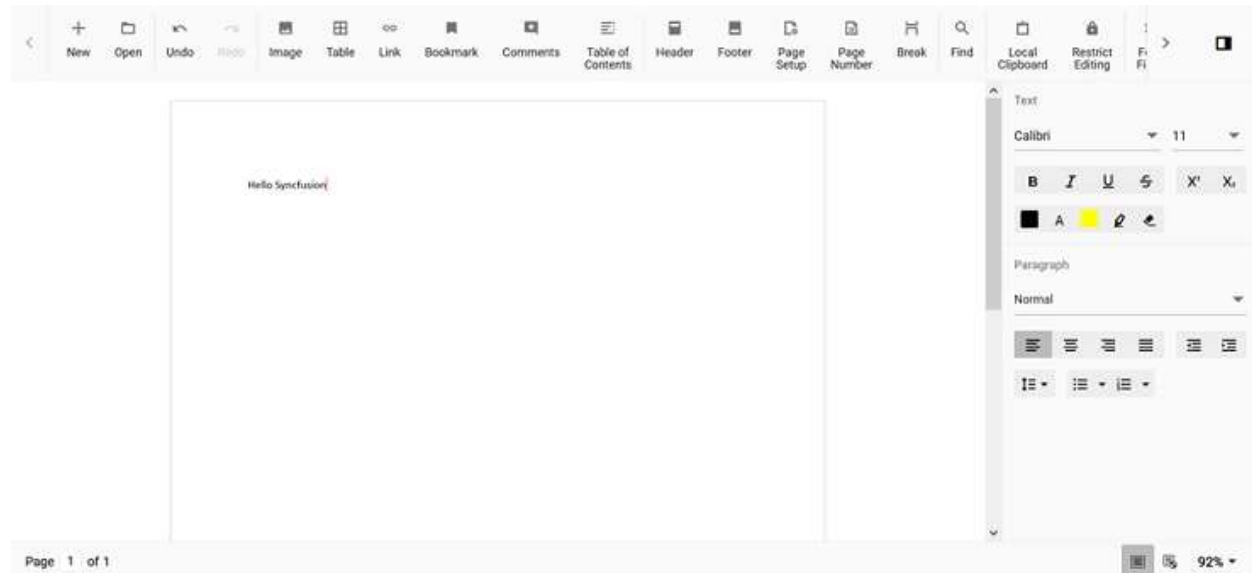
Note: The above example illustrates inserting HTML content. Similarly, you can insert any rich-text content by converting any of the supported file formats (DOCX, DOC, WordML, HTML, RTF) to SFDT.

[Change the cursor color in document editor in ##Platform_Name## Document editor control](#)
Document Editor default cursor color is black. The user can change the color by overriding the css property using class name. The Document editor cursor css have a class named `e-de-blink-cursor`.

Please refer the below code snippet to change the cursor color to red.

```
,
.e-de-blink-cursor {
border-left: 1px solid red!important;
}
,
```

Output will be like below:



Hide tool bar and properties pane in `##Platform_Name##` Document editor control

Document editor container provides the main document view area along with the built-in toolbar and properties pane.

Document editor provides just the main document view area. Here, the user can compose, view, and edit the Word documents. You may prefer to use this component when you want to design your own UI options for your application.

Hide the properties pane

By default, Document editor container has built-in properties pane which contains options for formatting text, table, image and header and footer. You can use [showPropertiesPane](#) API in [DocumentEditorContainer](#) to hide the properties pane.

The following example code illustrates how to hide the properties pane.

```
`ts
import { DocumentEditorContainer, Toolbar } from '@syncfusion/ej2-documenteditor';
DocumentEditorContainer.Inject(Toolbar);

let container: DocumentEditorContainer = new DocumentEditorContainer({ enableToolbar: true, height:
'590px', showPropertiesPane:false });
container.serviceUrl = 'https://services.syncfusion.com/js/production/api/documenteditor/';
```



```
container.appendTo('#container');
`
```

Note: Positioning and customizing the properties pane in Document editor container is not possible. Instead, you can hide the existing properties pane and create your own pane using public API's.

Hide the toolbar

You can use [enableToolbar](#) API in [DocumentEditorContainer](#) to hide the existing toolbar.

The following example code illustrates how to hide the existing toolbar.

```
`ts
import { DocumentEditorContainer, Toolbar } from '@syncfusion/ej2-documenteditor';

let container: DocumentEditorContainer = new DocumentEditorContainer({ enableToolbar: false, height:
'590px' });

container.serviceUrl = 'https://services.syncfusion.com/js/production/api/documenteditor/';
container.appendTo('#container');
`
```

See Also

- [How to customize the toolbar](#)

Insert text or image in table programmatically in `##Platform_Name##` Document editor control
Using Document editor API's, you can insert [text](#) or [image](#) in [table](#) programmatically based on your requirement.

Use [selection](#) API's to navigate between rows and cells.

The following example illustrates how to create 2*2 table and then add text and image programmatically.

```
`ts
import { DocumentEditorContainer, Toolbar } from '@syncfusion/ej2-documenteditor';

DocumentEditorContainer.Inject(Toolbar);

let container: DocumentEditorContainer = new DocumentEditorContainer({ enableToolbar: true, height:
'590px' });

container.serviceUrl = 'https://services.syncfusion.com/js/production/api/documenteditor/';
container.appendTo('#container');

// To insert the table in cursor position
container.documentEditor.editor.insertTable(2,2);

// To insert the image at table first cell
container.documentEditor.editor.insertImage("data:image/png;base64,iVBORw0KGgoAAAANSUHEUgAA
AAUAAAFCAyAAACNbybIAAAAHIEQVQI12P4
//8/w38GIAXDIBKE0DHxgljNBAAO9TXL0Y4OHwAAAABJRU5ErkJggg==");
```

```
// To move the cursor to next cell
moveCursorToNextCell();

// To insert the image at table second cell
container.documentEditor.editor.insertImage("data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAA
AAUAAAAFCAYAAACNbyblAAAAHEIEQVQI12P4
//8/w38GIAXDIBKE0DHxgljNBAAO9TXL0Y4OHwAAAABJRU5ErkJggg==");

// To move the cursor to next row
moveCursorToNextRow();

// To insert text in cursor position
container.documentEditor.editor.insertText('Text');

// To move the cursor to next cell
moveCursorToNextCell();

// To insert text in cursor position
container.documentEditor.editor.insertText('Text');

function moveCursorToNextCell() {
// To get current selection start offset
var startOffset=container.documentEditor.selection.startOffset;

// Increasing cell index to consider next cell
var cellIndex= parseInt(startOffset.substring(6, 7)) + 1;

// Changing start offset
startOffset = startOffset.substring(0, 6) + cellIndex.toString() + startOffset.substring(7,
startOffset.length);

// Navigating selection using select method
container.documentEditor.selection.select(startOffset, startOffset);
}

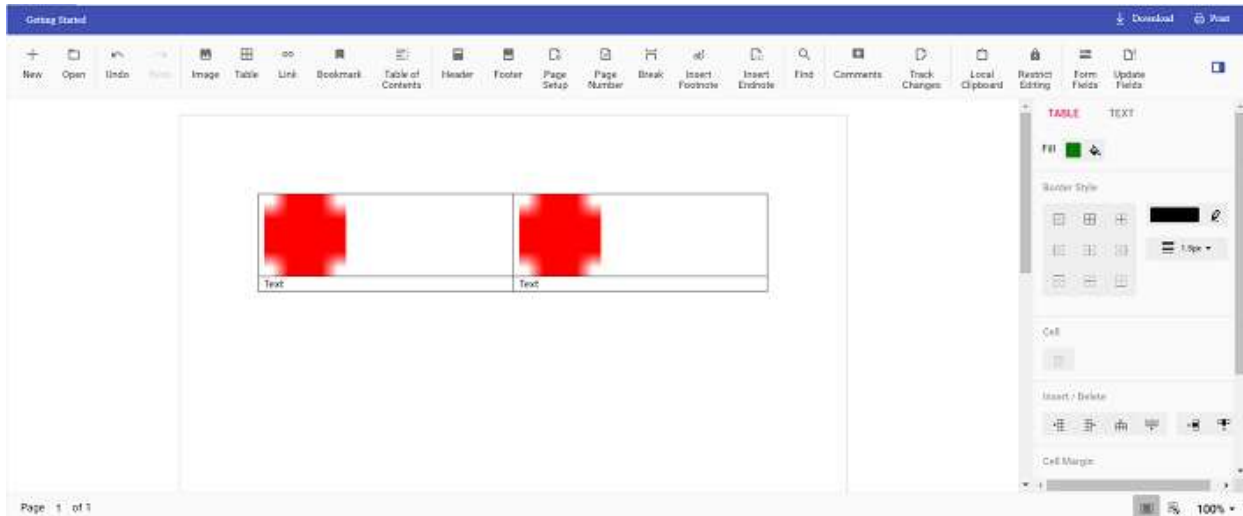
function moveCursorToNextRow() {
// To get current selection start offset
var startOffset=container.documentEditor.selection.startOffset;

// Increasing row index to consider next row
var rowIndex= parseInt(startOffset.substring(4, 5)) + 1;
var cellIndex= parseInt(startOffset.substring(6,7)) != 0? parseInt(startOffset.substring(6,7)) - 1:0;

// Changing start offset
startOffset = startOffset.substring(0, 4) + rowIndex.toString() + startOffset.substring(5, 6) + cellIndex +
startOffset.substring(7, startOffset.length);
```

```
// Navigating selection using select method
container.documentEditor.selection.select(startOffset, startOffset);
}
,
```

The output will be like below.



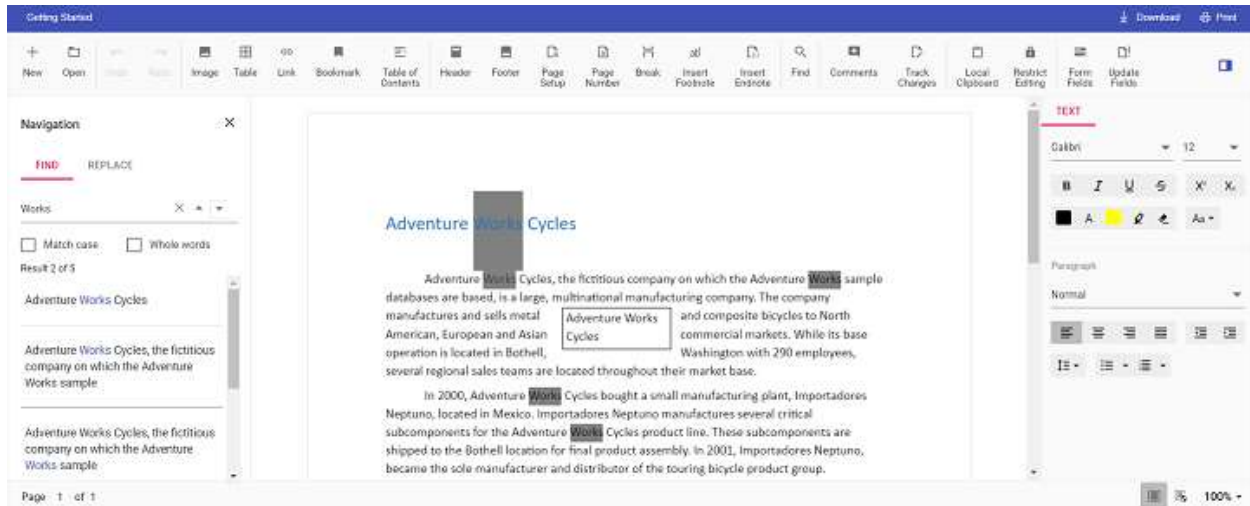
Change the default search highlight color in `##Platform_Name##` Document editor control. Document editor provides an options to change the default search highlight color using [searchHighlightColor](#) in Document editor settings. The highlight color which is given in [documentEditorSettings](#) will be highlighted on the searched text. By default, search highlight color is yellow.

Similarly, you can use [documentEditorSettings](#) property for DocumentEditor also.

The following example code illustrates how to change the default search highlight color.

```
`ts
let container: DocumentEditorContainer = new DocumentEditorContainer({ enableToolbar: true,height:
'590px',
// Add required search highlight color
documentEditorSettings: {
searchHighlightColor: 'Grey',
}
});
DocumentEditorContainer.Inject(Toolbar);
container.serviceUrl = 'https://services.syncfusion.com/js/production/api/documenteditor/';
container.appendTo('#container');
,
```

Output will be like below:



Disable auto focus in `##Platform_Name##` Document editor control

Document Editor gets focused automatically when the page loads. If you want the Document editor not to be focused automatically it can be customized.

The following example illustrates to disable the auto focus in DocumentEditorContainer.

```
`ts
```

```
let container: DocumentEditorContainer = new DocumentEditorContainer({ enableToolbar: true, height:
'590px', enableAutoFocus: false});
```

```
,
```

Note: Default value of [enableAutoFocus](#) property is `true`.

The following example illustrates to disable the auto focus in DocumentEditor.

```
`ts
```

```
let editor: DocumentEditor = new DocumentEditor({ height: '590px', enableAutoFocus: false});
```

```
,
```

Note: Default value of [enableAutoFocus](#) property is `true`.

Disable drag and drop in `##Platform_Name##` Document editor control

Document Editor provides support to drag and drop contents within the component and it can be customized(enable and disable) using [allowDragAndDrop](#) property in Document editor settings.

The following example illustrates to disable the drag and drop option in DocumentEditorContainer.

```
`ts
```

```
let container: DocumentEditorContainer = new DocumentEditorContainer({ enableToolbar: true, height:
'590px', documentEditorSettings: { allowDragAndDrop: false } });
```

```
,
```

Note: Default value of [allowDragAndDrop](#) property is `true`.

The following example illustrates to disable the drag and drop option in DocumentEditor.

`ts

```
let editor: DocumentEditor = new DocumentEditor({ height: '590px', documentEditorSettings: {
allowDragAndDrop: false } });
```

,

Note: Default value of [allowDragAndDrop](#) property is `true`.

Enable ruler

How to enable ruler in Document Editor component

Using ruler we can refer to setting specific margins, tab stops, or indentations within a document to ensure consistent formatting in Document Editor.

The following example illustrates how to enable ruler in Document Editor

INDEX.JS

```
//Initialize Document Editor component.
var documenteditor = new ej.documenteditor.DocumentEditor({
  isReadOnly: false, height: '370px', serviceUrl:
  'https://services.syncfusion.com/js/production/api/documenteditor/',
  documentEditorSettings: { showRuler: true }
});
//Enable all built in modules.
documenteditor.enableAllModules();
document.getElementById('container_ruler_button').addEventListener('click',
  function () {
    documenteditor.documentEditorSettings.showRuler =
    !documenteditor.documentEditorSettings.showRuler;
  });
//Render Document Editor component.
documenteditor.appendTo('#DocumentEditor');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
documenteditor/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="container_ruler_button">Show/Hide Ruler</button>
        <div id="DocumentEditor" style="height:420px">

            </div>
        </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

How to enable ruler in Document Editor Container component

Using ruler we can refer to setting specific margins, tab stops, or indentations within a document to ensure consistent formatting in Document Editor Container.

The following example illustrates how to enable ruler in Document Editor Container.

INDEX.JS

```

//Initialize Document Editor Container component.
var documenteditorcontainer = new
ej.documenteditor.DocumentEditorContainer({
    height: '590px', documentEditorSettings: { showRuler: true }
});
//Inject require modules.
ej.documenteditor.DocumentEditorContainer.Inject(
    ej.documenteditor.Toolbar
);
document.getElementById('container_ruler_button').addEventListener('click',
    function () {
        container.documentEditorSettings.showRuler =
!container.documentEditorSettings.showRuler;
    });
//Render Document Editor Container component.
documenteditorcontainer.appendTo("#DocumentEditor");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>

```

```

<title>EJ2 Animation</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
documenteditor/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/fabric.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="container_ruler_button">Show/Hide Ruler</button>
        <div id="DocumentEditor" style="height:420px">

        </div>
    </div>
</body>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

DropDown Menu

Icons in ##Platform_Name## Drop down button control

DropDownButton icons

DropDownButton can have an icon to provide the visual representation of the action. To place the icon on a DropdownButton, set the [iconCss](#) property to `e-icons` with the required icon CSS. By default, the

icon is positioned to the left side of the DropDownButton. You can customize the icon's position using the [iconPosition](#) property.

In the following example, the DropDownButton with default iconPosition and iconPosition as **Top** is showcased:

INDEX.JS

```
ej.base.enableRipple(true);
var items = [
  {
    text: 'Edit'
  },
  {
    text: 'Delete'
  },
  {
    text: 'Mark as Read'
  },
  {
    text: 'Like Message'
  }
];
//To position the icon to the left of the text on a DropDownButton.
var drpDownBtn = new ej.splitbuttons.DropDownButton({iconCss: 'ddb-icons e-
message', items: items, '#iconbutton'});
var drpDownBtn = new ej.splitbuttons.DropDownButton({iconCss: 'ddb-icons e-
message', items: items, iconPosition: 'Top', '#iconpstn'});
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 DropDownButton</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <button id="iconbutton">Message</button>
```



```

        <button id="iconpstrn">Message</button>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```
#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
@font-face {
font-family: 'e-db-icons';
src:
url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAwAgTlMvMj0jSRoAAAEoAAAAVmNtYXdDnFudgAAABkAAAADpnbHlmSrK
TCAAAAdgAAC4aGVhZBKtK8cAADQAAAAANmhoZWEHmQNtAAAArAAACRobXR4D7gAAAAAAAAAAAAA
QBgG9jYQB4ADoAAAHMAAAACmlheHABEAAYAAABCAAACBuYWllH00mDAAAAPAAAAJJCg9zdIwksr0
AAATcAAAAATQABAADUv9qAFoEAAAA//4D6gABAAAAAAAAAAAAAAAAAAAAAAAAABAABAAAAAQAGC/PS18
PPPUACwPoAAAAANfSc3wAAAAA19JzfAAAAAAD6gPqAAAAACAACAAAAAAAAAAAAAEAAAAEAAwAAgAAAA
AAgAAAAAoACgAAP8AAAAAAAAAAQPuAZAABQAAAAnoCvAAAAAIwCegK8AAAB4AAxAQTAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wPnBQNS/2oAWgPqAJYAAAAABAAAAAAAAABAAAAAPoAAA
D6AAAA+gAAAAAAAAIAAADAAAAFAADAAEEAAAAUAQAjgAAAAQABAABAADnBf//AADnA///AAAAQA
EAAAAAQACAAMAAAAAAAAAHAA6AFwAAAAACAAAAAAPqA2UABgAKAAA3IREjCQEjBRcBIQID6AL+Dv4
NAQEY3QG4/I+IASL+GAHonroBcwAAAAIAAAAAA8YD6gAFAAoAADChESMJASUHCQImA6AD/jL+MQE
EyWGWAZzb+agICX/4+AcLXsv6cAWQBZAAAAEAAAAA+oD6gALAAATCQEXCQE3CQEnCQECATP+zCI
BMgEzwf7OATLB/s3+zgMp/s3+zsIBM/7NwgEyATPB/s4BMgAAAAASAN4AAQAAAAAAAAABAAAAQA
AAAAAAQAKAAEAAQAAAAAAAgAHAAsAAQAAAAAAAwAKABIAAQAAAAAABAABAAQAAAAAABQALACY
AAQAAAAAABgAKADEAAQAAAAAACgAsAdSAAQAAAAAAcWASAGcAAwABBakAAAACAhkAAwABBakAAQA
UAHSAAwABBakAAgAOAI8AAwABBakAAwAUAJ0AAwABBakABAAUALEAAwABBakABQAWAMUAAwABBak
ABgAUANsAAwABBakACgBYAO8AAwABBakACwAkAUcgZS1kYi1pY29uc1JlZ3VsYXJlLWRiLWljbj25
zZS1kYi1pY29uc1Zlcnpjb24gMS4wZS1kYi1pY29uc0ZvbncgZ2VuzXJhdGVkIHVzaW5nIFN5bmN
mdXNpb24gTWV0cm8gU3RlZGlvd3d3LnN5bmNmZXNpb24uY29tACAazQatAGQAYgAtAGkAYwBvAG4
AcwBSAGUAzwBlAGWAYQByAGUALQBkAGIALQBpAGMAbwBuAHMAZQatAGQAYgAtAGkAYwBvAG4AcwB
WAGUAcgBzAGkAbwBuACAAMQAuADAazQatAGQAYgAtAGkAYwBvAG4AcwBGAG8AbgBOACAAZWBlAG4
AZQByAGEAdABLAGQAIBLAHMaaQBuAGcAIBTAHkAbgBJAGYAdQBzAGkAbwBuACAATQBLAHQAcgB
vACAAUwB0AHUAZABpAG8AdwB3AHcALgBzAHkAbgBJAGYAdQBzAGkAbwBuAC4AYwBvAG0AAAAAAgA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAEAQIBAwEEAAQUADG1lc3NhZ2UtbnwFfbpAtyZWFF
kLXVuclVhZAZkZWxldGUAAAAAAAA==) format('truetype');
font-weight: normal;
font-style: normal;
}
.ddb-icons {
```

```
font-family: 'e-db-icons' !important;
speak: none;
font-size: 55px;
font-style: normal;
font-weight: normal;
font-variant: normal;
text-transform: none;
line-height: 1;
-webkit-font-smoothing: antialiased;
-moz-osx-font-smoothing: grayscale;
}
.e-message::before {
  content: '\e703';
}
button {
  margin: 25px 5px 20px 20px;
}
```

Icon only DropDownButton

Icon only DropDownButton can be achieved by using [iconCss](#) property and to hide drop down arrow

e-caret-hide class is added using [cssClass](#) property.

INDEX.JS

```
ej.base.enableRipple(true);
var items = [
  {
    text: 'New tab'
  },
  {
    text: 'New window'
  },
  {
    text: 'New incognito window'
  },
  {
    separator: true
  },
  {
    text: 'Print'
  },
  {
    text: 'Cast'
  },
  {
    text: 'Find'
  }
];
var options = {
  items: items,
  iconCss: 'e-icons e-menu',
  cssClass: 'e-caret-hide'
};
var drpDownBtn = new ej.splitbuttons.DropDownButton(options);
drpDownBtn.appendTo('#icononly');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 DropDownButton</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <button id="icononly"></button>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
  visibility: hidden;
}
#loader {
  color: #008cff;
  height: 40px;
  left: 45%;
  position: absolute;
  top: 45%;
  width: 30%;
}
.e-menu::before {
  content: '\e984';
}

```

The Essential JS 2 provides a set of icons that can be loaded by applying `e-icons` class name to the element. You can also use third party icons on the DropDownButton using the `iconCss` property.

DropDownButton with sprite image

Sprite images can be loaded in DropDownButton instead of font icons using `iconCss` property.

In this following example, `e-image` class is added with background url of the sprite image along with X and Y positions. The `width` and

`height` of the element set as `32px`.

INDEX.JS

```
ej.base.enableRipple(true);
// Initialize action items.
var items = [
  {
    text: 'Display Settings'
  },
  {
    text: 'System Settings'
  },
  {
    text: 'Additional Settings'
  }
];
// Initialize DropDownButton options.
var options = {
  items: items,
  cssClass: 'e-caret-hide',
  iconCss: 'e-image'
};
// To initialize the DropDownButton with sprite image.
var drpDownBtn = new ej.splitbuttons.DropDownButton(options, '#icononly');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 DropDownButton</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="icononly"></button>
    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.e-btn-icon.e-image {
    background: url(./spritesheet.png) -384px -48px;
    width: 32px;
    height: 32px;
}

```

Vertical button

Vertical button in DropDownButton can be achieved by adding `e-vertical` class using [cssClass](#) property.

The following example illustrates how to provide vertical support in DropDownButton component.

INDEX.JS

```

ej.base.enableRipple(true);
var items = [
    {
        text: 'Edit'
    },
    {
        text: 'Delete'
    },
    {
        text: 'Mark as Read'
    },
    {
        text: 'Like Message'
    }
]

```

```

    }];
    //To position the icon to the left of the text on a DropDownButton.
    var drpDownBtn = new ej.splitbuttons.DropDownButton({iconCss: 'ddb-icons e-
message', items: items, cssClass: 'e-vertical', iconPosition: 'Top'},
    '#iconbutton');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 DropDownButton</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <button id="iconbutton">Message</button>
  </div>
  <script>
  var ele = document.getElementById('container');
  if(ele) {
    ele.style.visibility = "visible";
  }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
  visibility: hidden;
}
#loader {
  color: #008cff;
  height: 40px;
  left: 45%;
  position: absolute;
  top: 45%;
  width: 30%;

```

```

}
@font-face {
font-family: 'e-db-icons';
src:
url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMj0jSRoAAAEoAAAAVmNtYXDNFudgAAABkAAAAADpnbHlmSrK
TCAAAAdgAAAC4aGVhZBKtK8cAAADQAAAAANmhoZWEHmQNtAAAArAAAACRobXR4D7gAAAAAYAAAA
QbG9jYQB4ADoAAAHMAAAACm1heHABEAAAYAAABCAAAACBuYw1lH00mDAAAAPAAAAJJcG9zdIwksr0
AAATcAAAATQABAAADUv9qAFoEAAAA//4D6gABAAAAAAAAAAAAAAAAAAAAAAAAAABAAAAAAQAGc/PS18
PPPUACwPoAAAAANfSc3wAAAAA19JzfAAAAAAD6gPqAAAAACAACAAAAAAAAAAAAEAAAAEAAwAAgAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAAPuAZAABQAAAAnoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMMAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wPnBQNS/2oAWgPqAJYAAAAABAAAAAAAAABAAAAAPoAAA
D6AAAA+gAAAAAAIAAAADAAAAFAADAAEAAAAUAAQAjgAAAAQABAABADnBf//AADnA///AAAAQA
EAAAAAQACAAMAAAAAAAHAA6AFwAAAAACAAAAAAAPqA2UABgAKAAA3IREjCQEjBRcBIQID6AL+Dv4
NAQEY3QG4/I+IAsL+GAHonroBcwAAAAIAAAAAA8YD6gAFAAoAADchESMJASUHCQImA6AD/jL+MQE
EywGWAZb+agICX/4+AcLXsv6cAWQBZAAAAEAAAAAAAA+oD6gALAAATCQEXCQE3CQEnCQECATP+zcI
BMgEzwf7OATLB/s3+zgMp/s3+zsIBM/7NwgEyATPB/s4BMgAAAAASAN4AAQAAAAAAAAABAAAAQA
AAAAAQAKAAEAAQAAAAAAAAgAHAAsAAQAAAAAAAAAwAKABIAAQAAAAABAAKABwAAQAAAAABQALACY
AAQAAAAABgAKADEAAQAAAAAACgAsADsAAQAAAAACwASAGCAAwABBAkAAAACAHkAAwABBAkAAQA
UAHsAAwABBAkAAgAOAI8AAwABBAkAAwAUAJ0AAwABBAkABAAUALEAAwABBAkABQAWAMUAwABBAk
ABgAUANsAAwABBAkACgBYAO8AAwABBAkACwAkAUcgZS1kYi1pY29uc1JlZ3VsYXJlLWljb25
zZS1kYi1pY29uc1ZlcnNpb24gMS4wZS1kYi1pY29uc0ZvbnQgZ2VuZlZhdGVkIHVzaW5nIFN5bmN
mdXNpb24gTWV0cm8gU3R1ZGlvd3d3LnN5bmNmdXNpb24uY29tACAAZQAtAGQAYgAtAGkAYwBvAG4
AcwBSAGUAZwB1AGwAYQByAGUALQBkAGIALQBpAGMABwBuAHMAZQAtAGQAYgAtAGkAYwBvAG4AcwB
WAGUAcgBzAGkAbwBuACAAMQAuADAAZQAtAGQAYgAtAGkAYwBvAG4AcwBGAG8AbgB0ACAAZwB1AG4
AZQByAGEAdABlAGQAIABlAHMAaQBuAGcAIABTAHkAbgBjAGYAdQBzAGkAbwBuACAATQBlAHQAcgB
vACAAUwB0AHUAZABpAG8AdwB3AHcALgBzAHkAbgBjAGYAdQBzAGkAbwBuAC4AYwBvAG0AAAAAAgA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAEAQIBAwEEAQUADG1lc3NhZ2UtZWVpZAtyZWV
kLXVucmVhZAZkZWxldGUAAAAAAAA==) format('trueType');
font-weight: normal;
font-style: normal;
}
.ddb-icons {
font-family: 'e-db-icons' !important;
speak: none;
font-size: 55px;
font-style: normal;
font-weight: normal;
font-variant: normal;
text-transform: none;
line-height: 1;
-webkit-font-smoothing: antialiased;
-moz-osx-font-smoothing: grayscale;
}
.e-message::before {
content: '\e703';
}
button {
margin: 25px 5px 20px 20px;
}

```

See Also

- [Dropdown popup with icons](#)
- [Customized icon size](#)

Popup items in ##Platform_Name## Drop down button control

Icons

The popup action item have an icon or image to provide visual representation of the action. To place the icon on a popup item, set the [iconCss](#) property to e-icons with the required icon CSS. By default, the icon is positioned to the left side of the popup action item.

In the following sample, the icons for edit, delete, mark as read and like message menu items are added using the iconCss property.

INDEX.JS

```
ej.base.enableRipple(true);
var items = [
  {
    text: 'Edit',
    iconCss: 'ddb-icons e-edit'
  },
  {
    text: 'Delete',
    iconCss: 'ddb-icons e-delete'
  },
  {
    text: 'Mark as Read',
    iconCss: 'ddb-icons e-read'
  },
  {
    text: 'Like Message',
    iconCss: 'ddb-icons e-like'
  }
];
//To position the icon to the left of the text on a DropDownButton.
var drpDownBtn = new ej.splitbuttons.DropDownButton({iconCss: 'ddb-icons e-
message', items: items}, '#iconbutton');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 DropDownButton</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
```



```
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="iconbutton">Message</button>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

STYLES.CSS

```
#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
@font-face {
    font-family: 'ddb-icons';
    src:
        url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMjltSfYAAAEoAAAAVmNtYXNnG0dnAAABmAAAAD5nbHlm/RE
9ZwAAAAegAAAJ8aGVhZBOPuxsAAADQAAAAANmhoZWEIUQQHAAAArAAAACRobXR4GAAAAAAAAAYAAAA
YbG9jYQHiAO4AAAHYAAAAADm1heHABFACZAAABCAAAACBuYW11l1LBM9QAABGQAAAI9cG9zdOdmKCA
AAAAkAAAAZgABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAABgABAAAAAQAAfi9ISf8
PPPUACwQAAAAANg+uxUAAAAA2D67FQAAAAAD9AP0AAAACAACAAAAAAAAAAAAEAAAAGAI0ABAAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQQA ZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnBAQAAAAAXAQAAAAAAAAABAAAAAAAAABAAAAQAAAA
EAAAABAAAAAQAAAAEAAAAAAAAGAAAAAMAAAAUAMAAQQAABQABAAqAAAAABAEAAEAAOCe//8AAOc
A//8AAAAABAAQAAAAABAAIAAwAEAAUAAAAAAAAAALgBaAHYAlAE+AAAAAwAAAAAD9AP0AAIABgAZAAA
3JSc3FwEnNwcXPwM1LwcPAGwBJOo76QHT6qlu6XIFBAICBAWmCAkJCgkJCQw66jrpAdLpqW7pcgg
JCgKQCQimBwQDAQEEDBAAAAAAEAAAAAANNA/QAAwAHABAAGAAAAAREjESMRIxEnETMVIITUzESE3IxU
hNSM1IQLIh4SFhUICGED9ZoWFApqF/nACp/3qAhb96gIWQv1mQ0MC3YVCQkMAAAAAAGAAAAAD8wN
uAAYACgAANYeERIwKBiWUXASEMA+gC/g3+DgEBGNwBufyOkgLC/hcB6Z+5AXIAAAACAAAAAAPQA/Q
ABQAKAAA3IREjCQELBwkCMAOGA/4x/jIBA8sBlgGX/moMA17+PgHC2LL+nAfKAwQAAAACAAAAAAP
0A8UAAwCMAAA3MxEjAQ8DFRcPDBEzNx8ENxc/Cj0BLwU/Cy8INzU/CDUvBTU/DTUvCQc1PwQ1Lws
jDwEMra0B+QIKBAEBAQEYIREREHMiCQkoEAYhBzUHHjmt2w4FCAsNCwkFAwQCAGQJBgIBAQEEDdgQ
JCAYHAWMBAQEBAwMDCQIBAQMwCwUEBAMDAgICBAQKAQEBAaOHBWYFBQQDAwEBAQEEDBQcJBQUFBhH
+rQ8JBAMCAQEDAwoMFQMHBgwLDQcHWgGHAd4BBQMDdh8KBCw6HRScGi8JCBsM/ooBAR8DAQEBAGF
BAWYKCGwGCAGIBQgJCAsFBAQEBOQMGawCICAwIBwgHBgYGBQUJBAIGAGQMCQYFBgcJCQoJCAGHCwQ
CBQMCABAQEBOQUGBwcIBWYGBgYKQCgGAGIBAQEBRjEZGhsNDQwNcyIeMQQEAGQBAQIAAAASAN4AAQA
AAAAAAAAABAAAAAQAAAAAAQAJAEEAAQAAAAAAAGAAAAoAAQAAAAAAAwAJABEAAQAAAAABAAJABO
AAQAAAAAABOALACMAAOAAAAAABGqAJAC4AAQAAAAAACqAsADcAAQAAAAAACwASAGMAAwABBAKAAAA
```

```

CAHUAaWABBAkAAQASAHcAAwABBAkAAgAOAIkAAwABBAkAAwASAJcAAwABBAkABAASAKkAAwABBAk
ABQAWALsAAwABBAkABgASANEAAwABBAkACgBYAOMAAwABBAkACwAkATsgZGRiLWljb25zUmVndWx
hcmRkYi1pY29uc2RkYi1pY29uc1ZlcnNpb24gMS4wZGRiLWljb25zRm9udCBnZW5lcmF0ZWQgdXN
pbmcgU3luY2Zlc2lubiBNZXRYbyBTdHVkaW93d3cuc3luY2Zlc2lubi5jb20AIABkAGQAYgAtAGk
AYwBvAG4AcwBSAGUAZwB1AGwAYQByAGQAZABiAC0AaQBjAG8AbgBzAGQAZABiAC0AaQBjAG8AbgB
zAFYAZQByAHMAaQBvAG4AIAAxAC4AMABkAGQAYgAtAGkAYwBvAG4AcwBGAG8AbgB0ACAAZwB1AG4
AZQByAGEAdABlAGQAIAB1AHMAaQBuAGcAIABTAHkAbgBjAGYAdQBzAGkAbwBuACAATQBlAHQAcgB
vACAAUwB0AHUAZABpAG8AdwB3AHcALgBzAHkAbgBjAGYAdQBzAGkAbwBuAC4AYwBvAG0AAAAAaG
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAQAQIBAwEEAQUBBgEHAAdlZG10XzAzCWRlbGV
0ZV8wMgxtZXNzYWdlLW1haWwLcmVhZC11bnJlYWQJbGlrZS0tLTAAxAAAAA==)
format('true');
    font-weight: normal;
    font-style: normal;
}
.ddb-icons {
    font-family: 'ddb-icons' !important;
    speak: none;
    font-size: 55px;
    font-style: normal;
    font-weight: normal;
    font-variant: normal;
    text-transform: none;
    line-height: 1;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
}
.e-message::before {
    content: '\e702';
}
.e-edit::before {
    content: '\e700';
}
.e-delete::before {
    content: '\e701';
}
.e-read::before {
    content: '\e703';
}
.e-like::before {
    content: '\e704';
}
button {
    margin: 25px 5px 20px 20px;
}

```

Navigations

Actions in DropDownButton can be used to navigate to the other web page when action item is clicked. This can be achieved by providing link to the action item using `url` property.

In the following sample, navigation URL for Flipkart, Amazon, and Snapdeal action items are added using the `url` property:

INDEX.JS

```

ej.base.enableRipple(true);
var items = [

```

```

    {
        text: 'Flipkart',
        iconCss: 'e-cart-icon e-link',
        url: 'https://www.google.co.in/search?q=flipkart'
    },
    {
        text: 'Amazon',
        iconCss: 'e-cart-icon e-link',
        url: 'https://www.google.co.in/search?q=amazon'
    },
    {
        text: 'Snapdeal',
        iconCss: 'e-cart-icon e-link',
        url: 'https://www.google.co.in/search?q=snapdeal'
    }
    ]];
var menuOptions = {
    items: items,
    iconCss: 'e-cart-icon e-shopping',
    beforeItemRender: (function (args) {
        args.element.getElementsByTagName('a')[0].setAttribute('target',
'_blank');
    })
};
var drpDownBtn = new ej.splitbuttons.DropDownButton(menuOptions, '#action');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 DropDownButton</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="action">Shop By</button>
    </div>
<script>
var ele = document.getElementById('container');
```

```
if(ele) {
    ele.style.visibility = "visible";
}

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

STYLES.CSS

```
#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
@font-face {
font-family: 'cart';
src:
url(data:application/x-font-ttf;charset=utf-
8;base64,AEEAAAAKAIAAAwAgTlMvMj0gSQ4AAAEoAAAAMVnTYXDnEodVAAABiAAAAADZnbHlmGat
ngwAAACgAAADYaGVhZBKTpP4wAADQAAAAANmhoZWEHMqNpAAAArAAAACRobXR4B+j//gAAAAyAAAAA
IbG9jYQBsaAAAAAHAAAAABmlheHABDwbQAAABCAAAACBuYW1lfiv2lQAAAqAAAAIBcG9zdIZzcJA
AAASkAAAAOgABAAADUv9qAfOEAP/+//wD7AABAAAAAAAAAAAAAAAAAAAAAGABAAAAAQAA2UwSaF8
PPPUACwPoAAAAANfSFWUAAAAA19J9Zf/+AAAD7APdAAAAACAACAAAAAAAAAAAAEAQAQAAwAAAAA
AAgAAAAoAcGAAAP8AAAAAAAAAAAAQP0AZAABQAAAnoCvAAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnAANS/2oAWGpPdAJYAAAAABAAAAAAAAABAAAAAPo//4
AAAAACAAAAAwAAABQAAwABAAAAFAAEACIAAAAEAAQAAQAA5wD//wAA5wD//wAAAAEABAAAAAEAAAA
AAAAAbAAAAAP//gAAA+wD3QAJABIAQWAAJR4BMjY0JicOAQUeATI2NCYiBgEOAwclIgYXEx4BMwU
yFgcOAQCllIgYXBhYXBT4BPwE2PwI2NxM+AyCMiyIGAeABJzonJx0gJ/6jASc6Jyc6JwMXIDgZPyX
9giQkBKIIOyQBacQGcQo/Jp7RIxcBARcjAVgkQg0QDgkICgkNkw4xBwCBScJE1UdJyc6JwEBJx0
dJyc6JycDZQg0PigBAy0k/uAkMAQnHRsmAQMTDA8QAQMBlCILihYWFxYhAYciNg4YDBMCAAAAEgD
eAAEAAAAAAAAAAAAQAAAAEAAAAAAAAEABABAAEAAAAAAAAIABwAFAAEAAAAAAAAAMABAAMAAEAAAAAAQ
ABAAQAAEAAAAAAAAUACWAUAEEAAAAAAAAAYABAafAAEAAAAAAAAAoALAAjAAEAAAAAAAAAsAEGBPAAMAAQQ
JAAAAAGbhAAMAAQQJAAEACABjAAMAAQQJAAAIADgBrAAMAAQQJAAAMACAB5AAMAAQQJAAQACACBAAM
AAQQJAAUAFgCJAAMAAQQJAAAYACACfaAMAAQQJAAoAWACnaAMAAQQJAAAsAJAD/IGNhcnRSZWdlbGF
yyY2FydGNhcnRWZXJzaW9uIDEuMGNhcnRGRz250IGdlbmVyYXRlZCB1c2luZyBTeW5jZnVzaW9uIEI
ldHJVIFN0dWRpb3d3dy5zeW5jZnVzaW9uLmNvbQAgAGMAYQByAHQAUGblAGcAdQBsaGEAcgBjAGE
AcgBOAGMAYQByAHQAVgBlAHIAcwBpAG8AbGAgADEALG AwAGMAYQByAHQARgBvAG4AdAAgAGcAZQB
uAGUAcgBhAHQAZQBkACAAdQBzAGkAbgBnACAAUwB5AG4AYwBmAUAUAcwBpAG8AbGAgAE0AZQB0AHI
AbwAgAFMAdABlAGQAaQBvAHcAdwB3AC4AcwB5AG4AYwBmAUAUAcwBpAG8AbGAuAGMAbwBtAAAAAAI
AAAAAAAAAACgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAgECAQMAEHNNob3BwaW5nLWNhcnQtMDUAAAA
A) format('trueType');
font-weight: normal;
font-style: normal;
}
.e-cart-icon {
    font-family: 'cart' !important;
    speak: none;
    font-size: 55px;
    font-style: normal;
    font-weight: normal;
```

```
font-variant: normal;
text-transform: none;
line-height: 1;
-webkit-font-smoothing: antialiased;
-moz-osx-font-smoothing: grayscale;
}
.e-shopping::before {
  content: '\e700';
}
.e-link::before {
  content: '\e700';
}

button {
  margin: 25px 5px 20px 20px;
}
```

Template

Item templating

Popup items can be customized using the [beforeItemRender](#) event. The item render event triggers while rendering each popup action item. The event argument will be used to identify the action item and customize based on the requirement.

The following popup template is customized using [beforeItemRender](#) event by appending [span](#) and [div](#) element on each [li](#) rendering:

INDEX.JS

```
ej.base.enableRipple(true);
var items = [
  {
    text: 'Edit'
  },
  {
    text: 'Cut'
  }
];
var ddbOption = {
  iconCss: 'e-ddb-icons e-paste',
  cssClass: 'e-vertical',
  items: items,
  iconPosition: 'Top',
  beforeItemRender: itemRender
}
var drpDownBtn = new ej.splitbuttons.DropDownButton(ddbOption,
'#iconbutton');
function itemRender(args) {
  if (args.item.text === 'Edit') {
    args.element.innerHTML = '<span></span><div><b>Paste  
Text</b><div>Provides option to paste only the<br>selected  
text.</div></div>';
    args.element.style.height = '80px';
    var span = args.element.children[0];
    span.setAttribute('class', 'e-cm-icons e-pastetext e-align');
    var div = args.element.children[1];
    div.setAttribute('class', 'e-div-align');
```

```

    } else {
        args.element.innerHTML = '<span></span><div><b>Paste
Special</b><div>Provides options to paste formulas,<br> values, comments,
validations etc...</div></div>';
        args.element.style.height = '80px';
        var span = args.element.children[0];
        span.setAttribute('class', 'e-cm-icons e-pastespecial e-align');
        var div = args.element.children[1];
        div.setAttribute('class', 'e-div-align');
    }
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 DropDownButton</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="styles.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="iconbutton">Paste</button>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

/* csslint ignore:start */
@font-face {
    font-family: 'e-context-menu';
    src:

```

```
url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMjJNRVMAAAEoAAAAVmNtYXNDicOK6AAABjAAAAAHnHlMhGcE
PFQAAACwAAAMWagVhZA69CA8AADQAAAAANmhoZWEH9AQEAAAArAAAACRobXR4DAAAAAAAAAYAAAA
MbG9jYQDYAZgAAAHEAAAAACG1heHABEGDAAAABCAAAACBuYw1lY1dlQAABPwAAAKFcG9zdPjWcMo
AAAEeAAAAASAABAAAEAAAAAFwEAAAAAADlwABAAAAAAAAAAAAAAAAAAAAAAwABAAAAAQAAgmhm8l3
PPPUACwQAAAAAANYD4Y8AAAAA1gPhjwAAAAADlwP0AAAAACAACAAAAAAAAAAAAEAAAAADALQABQAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQQAQAAZAAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA4mDiYQQAAAAAXAQAAAAAAAAABAAAAAAAABAAAAAQAAAA
EAAAAAAAAAAgAAAAAMAAAUAMAAQAABQABAakAAAAABAAEAAEAAOJh//8AAOJg//8AAAABAAQAAAA
BAAIAAAAAANgBmaAFAAAAAAOXA/QABAALAC0ATgCzAAABISCHJzcVHwc/BY8HDwYBFSE1MxEhESU
HFQ8GLwc/Bx8GJYsBDw4RHW4zITM/DhEvDisBLw4raAQ8NAUQBdlxAFr0BAwQGBwgICGkJCACGBAM
BAQMEBgICQKkCAGHBgQD/qYB1l79jQFoAQMEBgHCQkJCQqGBGQDAQEDBAYGCAkJCQkHBwYEA6y
9CgkICQqHBwCGBQQAEMBAQEBAwMDBQUGBwCHCAkJCQoCeAoJCAkJBwCHBgUEBAMDAQEBAQMDAwU
FBGCHBwgJCAkJvQqEBgUHBwCICQkJCgoKCwsLCwoKCgkJCQqHBwCfBgQBBYVRuh0FBQkIBwUFAgE
BAgUFBwgJCGkJCACGBAMBAQMEBgICQEifX39LwLRMwQFCAGHBQUCAQECEBQUHCAgJCQkIBwUEAwE
BAwQFBwgJIGICAwQFBQYGBwgICAKJCF0pCQkJCAGIBWYGBQUEAwICAGIDBAUFBgYHCAgICQkJAtC
JCQkICAGHBgYFBQQDAgIKCQKICAGHBgYFBAQDAgICAGMEBAUGBgICAGJCQAFAAAAAAOXA/QABwA
PABcAOACdAAABHwIjPwIDMzcZfZMDIycVITUzESERJQcVDwYvBz8HHwYnKwEPdhEfDjMhMz8OES8
OKwEvDisBDw0B/wQKK3MmBQ6dMyeHKDWCO90B1l79jQFoAQMEBgHCQkJCQqGBGQDAQEDBAYGCAkJ
JCQkHBwYEA6y9CgkICQqHBwCGBQQAEMBAQEBAwMDBQUGBwCHCAkJCQoCeAoJCAkJBwCHBgUEBAM
DAQEBAQMDAwUFBGCHBwgJCAkJvQqEBgUHBwCICQkJCgoKCwsLCwoKCgkJCQqHBwCfBgQCFReigG4
SM/6wd3cBe/t9ff0vAtEzBAUICAcFBQIBAQIFBQcICAKJCQqHBGQDAQEDBAUHCAkJAgIDBAUFBgY
HCAgICQkJ/SkJCQkICAGHBgYFBQQDAgICAGMEBQUGBgICAGJCQkC1wkJCQgICAcGBgUFBAMCAgo
JCQgICAcGBgUEBAMCAgICAwQEBQYGBwgICAKJAAAAABIA3gABAAAAAEEEEAAAAABAAAAAABAA8
AAQABAAAAAAACAACAEABAAAAAADAA8AFwABAAAAAEEEEAA8AJgABAAAAAFAAAsANQABAAAAAA
GAA8AQABAAAAAAKACwATwABAAAAAALABIAewADAAEECQAAAAIAjQADAAEECQABAB4AjwADAAE
ECQACAA4ArQADAAEECQADAB4AuWADAAEECQAEAB4A2QADAAEECQAFABYA9wADAAEECQAGAB4BDQA
DAAEECQAKAFgBKwADAAEECQALACQBgyBDb250ZXh0TWVudSAoMilSZWdlbGFGYQ29udGV4dE1lbnU
gKDIpQ29udGV4dE1lbnUgKDIpVmVyc2lubiAxLjBDb250ZXh0TWVudSAoMilGb250IGdlbmVyYXR
lZCB1c2luZyBTeW5jZnVzaW9uIE1ldHJvIFN0dWRpb3d3dy5zeW5jZnVzaW9uLmNvbQAgAEMAbwB
uAHQAZQB4AHQATQBlAG4AdQAgACgAMgApAFTIAZQBnAHUAbABhAHIAQwBvAG4AdABlAHgAdABNAGU
AbgBlACAaKAAYAcKAwBvAG4AdABlAHgAdABNAGUAbgBlACAaKAAYAcKAVgBlAHIAcWBPAG8AbgA
gADEALgAwAEMAbwBuAHQAZQB4AHQATQBlAG4AdQAgACgAMgApAEYAbwBuAHQAIABnAGUAbgBlAHIA
YQYQB0AGUAZAAGAHUAcwBpAG4AZWAgAFMAEQBuAGMAZgBlAHMAAQBuAG4ALGBlAG8AbQAAAAACAAAAA
AAAOAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAMBAgEDAQQAD01UX1Bhc3RlU3BlY2lhbAAXNVF9QYXN
0ZVRleHQAAA==) format('trueType');
font-weight: normal;
font-style: normal;
}
/* csslint ignore:stop */
.e-cm-icons {
font-family: 'e-context-menu';
font-style: normal;
font-variant: normal;
font-weight: normal;
line-height: 1;
text-transform: none;
}
@font-face {
font-family: 'ddb-icons';
src:
url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMjJ0gSRKAAAEoAAAAVmNtYXNDne+dkAAABlAAAAADxnbHlmlh3
3NQAAAdwAAAJMaGvHbZBKOK9sAADQAAAAANmhoZWEHeANwAAAArAAAACRobXR4E6AAAAAAAAAYAAAA
UbG9jYQGOAegAAAAHQAAAADG1heHABEWBlAAABCAAAACBuYw1lLlBM9QAABCGAAAI9cG9zdMjntbU
AAAZoAAAAUAABAAADUv9gAFoEAAAAAADYqABAAAAAABAAAAAAAAAAAAAAAAABQABAAAAQAAAJXaQ18
```

```

PPPUACwPoAAAAANfSc4gAAAAA19JziAAA//oDyGPsAAAAACAACAAAAAEEEEAAFAFkABAAAAAA
AAgAAAAoACgAAAP8AAAAAQAAPtAZAABQAAAnoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnAwNS/2oAWGPsAJYAAAABAAAAAABAAAAAPoAAA
D6AAAA+gAAAPoAAAAAACAACAAAwAAABQAAwABAAAAFAAEACgAAAAEAAQAAQAA5wP//wAA5wD//wA
AAAEABAAAAEEAAGADAAQAAAAAAI4AwgEAASYAAwAA//oDNQPsAA4AHQBYAAALHgEOAScmJy4BNz4
BMzIFFgYHBgcGLgE2NzYzMhYBHgEXDgEHDgEHDgIWFxYXFjY3NjQ3PgE3HgEXFhQXHgE3PgE3PgE
uAScuAScuASc+ATc+AQcLASYWAVEfFx06IBkNCQIHCy8bCQG9BwIJDkrgOhoXHwoKgi/+TRlRDyE
OIxo+ExckFAQMfIkWvhCMBwYlFRYkBWCMF1YwFCALDAQUIxcUPhojDiAOUR4cAQvEwWSB6gtDTyc
JCBsSKxYhJ0gWKxIaCQknUEILAYcCf2TPI0w2HBUMdg0sOZsaKQ4ONZcniyYXNBgYNBcmiyc3OA8
GHRQaOzssDQ4mFRw2TiLOZGdBA/5vAZEDQQAEEAAAAAQa+kABQANABCAHwAAARUzFSErAYERIZU
jNSEBIREhESMVITUjMyMVITUjNSMC733+IT8B9D4+/oj+igE4AXc//c4++j8BOT+7AbZ8+gF2/ks
Bdz4//ksB9AF2fHw+Pj8AAAIAAAAA7cD6QACACQAAAEhEwMOAQcVITUmJyY1ND8BIRcWFxYVFAc
GKwEVITUmJyYnASMCKP8AguQrOy0BGkIRHREkASstEgEEDhQxEGGaJxUcLP7PDAFNAVL+PHBHCBS
bBgsUKR8wX3owBg4NFgsQGxsDFx1zAyMAAAACAAAAAPKA+oAAgATAAABFxEbDgEHHgEXETMRMxE
zETM1IQL+zPlabpADA5t0f2F+XP41AfBMAZgBJwmYCHSbA/48A2r8lgNqfgAAAAASAN4AAQAAAAA
AAAAABAAAAQAAAAAAQAJAEEAAQAAAAAAAgAHAAoAAQAAAAAAAwAJABEAQAAAAAABAAJABoAAQA
AAAAABQALACMAAQAAAAABgAJAC4AAQAAAAAACGAsADcAAQAAAAAACwASAGMAAwABBAkAAAAACAHU
AAwABBAkAAQASAHCAAwABBAkAAgAOAIkAAwABBAkAAwASAJcAAwABBAkABAASAKkAAwABBAkABQA
WALSAAwABBAkABgASANEAAwABBAkACgBYAOMAawABBAkACwAkATsgZGRiLWljB25zUmVndWxhcmR
kYilpY29uc2RkYilpY29uc1ZlcnNpb24gMS4wZGRiLWljB25zRm9udCBnZW5lcmF0ZWQgdXNpbmc
gU3luY2Zlc2lubiBNZXRYbyBTdHVkaW93d3cuc3luY2Zlc2lubi5jb20AIABkAGQAYgAtAGkAYwB
vAG4AcwBSAGUAZwB1AGwAYQByAGQAZABiAC0AaQBjAG8AbgBzAGQAZABiAC0AaQBjAG8AbgBzAFY
AZQByAHMAaQBvAG4AIAAxAAC4AMABkAGQAYgAtAGkAYwBvAG4AcwBGAG8AbgB0ACAAZwB1AG4AZQB
yAGEAdABlAGQAIABlAHMAaQBvAG4CAIABTAHkAbgBjAGYAdQBzAGkAbwBuACAATQBlAHQAcgBvACA
AUwB0AHUAZABpAG8AdwB3AHcALgBzAHkAbgBjAGYAdQBzAGkAbwBuAC4AYwBvAG0AAAAAAGAAAAA
AAAAKAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAFAQIBAwEEAQUBBgADY3V0CHBhc3RlXzAxBGZvbnQ
OcGFyYS1tYXJrLS0tMDMAAA==) format('trueType');
font-weight: normal;
font-style: normal;
}
.e-ddb-icons {
  font-family: 'ddb-icons' !important;
  speak: none;
  font-size: 55px;
  font-style: normal;
  font-weight: normal;
  font-variant: normal;
  text-transform: none;
  line-height: 1;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}
#container {
  visibility: hidden;
}
#loader {
  color: #008cff;
  height: 40px;
  left: 45%;
  position: absolute;
  top: 45%;
  width: 30%;
}
.e-pastespecial::before {
  content: '\e260';
}
.e-pastetext::before {

```



```

    content: '\e261';
  }
  .e-paste::before {
    content: '\e701';
  }
  button {
    margin: 25px 5px 20px 20px;
  }
  .e-dropdown-popup ul {
    max-width: 400px;
    white-space: nowrap;
  }
  .e-align {
    float: left;
    width: 15%;
    margin-top: 15px;
    font-size: 45px;
    color: grey;
  }
  .e-div-align {
    float: right;
    width: 75%;
    line-height: 23px;
    margin: 0 15px 0 0;
  }
}

```

Popup templating

The whole popup can be customized as per the requirement and it can be customized by handling it in [target](#) property.

In the following sample, the whole popup item is customized as table template by giving **div** as target and it can be achieved

using **target** property.

INDEX.JS

```

ej.base.enableRipple(true);
var menuOptions = {
  target: '#target',
  iconCss: 'e-icons e-table',
  iconPosition: 'Top',
  cssClass: 'e-vertical'
};
var drpDownBtn = new ej.splitbuttons.DropDownButton(menuOptions, '#action');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 DropDownButton</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="target" style="border: 1px solid grey;">
            <table>
                <caption style="height: 18px; background-color: #e0e0e0;"><b>Insert
Table</b></caption>
                <tbody><tr class="e-row"><td class="e-cell"></td><td class="e-
cell"></td><td class="e-cell"></td><td class="e-cell"></td><td class="e-
cell"></td><td class="e-cell"></td></tr>
                <tr class="e-row"><td class="e-cell"></td><td class="e-
cell"></td><td class="e-cell"></td><td class="e-cell"></td><td class="e-
cell"></td><td class="e-cell"></td></tr>
                <tr class="e-row"><td class="e-cell"></td><td class="e-
cell"></td><td class="e-cell"></td><td class="e-cell"></td><td class="e-
cell"></td><td class="e-cell"></td></tr>
                <tr class="e-row"><td class="e-cell"></td><td class="e-
cell"></td><td class="e-cell"></td><td class="e-cell"></td><td class="e-
cell"></td><td class="e-cell"></td></tr>
                <tr class="e-row"><td class="e-cell"></td><td class="e-
cell"></td><td class="e-cell"></td><td class="e-cell"></td><td class="e-
cell"></td><td class="e-cell"></td></tr>
                <tr class="e-row"><td class="e-cell"></td><td class="e-
cell"></td><td class="e-cell"></td><td class="e-cell"></td><td class="e-
cell"></td><td class="e-cell"></td></tr>
                <tr class="e-row"><td class="e-cell"></td><td class="e-
cell"></td><td class="e-cell"></td><td class="e-cell"></td><td class="e-
cell"></td><td class="e-cell"></td></tr>
            </tbody></table>
            </div>
            <button id="action">Table</button>
        </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```
#container {
```

```

        visibility: hidden;
    }
    #loader {
        color: #008cff;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
    .e-cell {
        border: 1px solid grey;
        padding: 8px;
    }
    .e-table::before {
        content: '\e705';
    }
    .e-row {
        padding-left: 3px;
        padding-right: 3px;
    }

    button {
        margin: 25px 5px 20px 20px;
    }

```

Separator

The Separators are the horizontal lines that are used to separate the popup items. You cannot select the separators. You can enable separators to group the popup items using the `separator` property.

In the following sample, cut, copy, and paste popup items are grouped using the `separator` property:

INDEX.JS

```

ej.base.enableRipple(true);
var items = [
    {
        text: 'Cut',
        iconCss: 'e-db-icons e-cut'
    },
    {
        text: 'Copy',
        iconCss: 'e-icons e-copy'
    },
    {
        text: 'Paste',
        iconCss: 'e-db-icons e-paste'
    },
    {
        separator: true
    },
    {
        text: 'Font',
        iconCss: 'e-db-icons e-font'
    },
    {

```

```

        text: 'Paragraph',
        iconCss: 'e-db-icons e-paragraph'
    }];
    var drpDownBtn = new ej.splitbuttons.DropDownButton({iconCss: 'e-icons e-edit', items: items}, '#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 DropDownButton</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <button id="element">Clipboard</button>
  </div>
  <script>
    var ele = document.getElementById('container');
    if(ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

STYLES.CSS

```

/* csslint ignore:start */
@font-face {
  font-family: 'e-dropdown-btn';
  src:
    url(data:application/x-font-ttf;charset=utf-8;base64,AAEAAAAKAIAAAwAgTlMvMjJNRVMAAAEoAAAAVmNtYXNDicOK6AAABjAAAADhnbHlmGcEPFQAAAcwAAAMwaGVhZA69CA8AAADQAAAAANmhoZWEH9AQEAAAArAAAACRobXR4DAAAAAAAAAYAAAAAMbG9jYQDYAZgAAAHEAAAACG1heHABEGDAAAABCAAAACBuYW11xY1d1QAABPwAAAKFcG9zdPJwcMoAAAEAAAAASAABAAAEAAAAAFwEAAAAAADlwABAAAAAAAAAAAAAAAAAAAAAwABAAAAAQAAgmhm8l8PPPUACwQAAAAAANYD4Y8AAAAA1gPhjwAAAAADlwP0AAAACAACAAAAAAAAAAAAEAAAADALQABQAAAAA
```

```

AAgAAAAoACgAAAP8AAAAAQAQAAZAAABQAAAOkCzAAAAAT8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA4mDiYQAAAAAXAQAAAAAAAAABAAAAAAAABAAAAAQAAAA
EAAAAAAAAAgAAAAAAAUAMAAQAABQABAaKAAAABAAEAAEAAOJh//8AAOJg//8AAAAABAAQAAAA
BAATIAAAAAANgBmAAFAAAAAAAXA/QABAA1AC0ATgCzAAABISchJzcVHwc/By8HDwYBFSE1MxEhESU
HFQ8GLwc/Bx8GJysBDw4RHw4zITM/DhEvDisBLw4rAQ8NAUQBdlxAfr0BAwQGBwgICgkJCACGBAM
BAQMEBgICQkKCAgHBgQD/qYB1l79jQFoAQMEBgHCQkJCQgGBgQDAQEDBAYGCAkJCQkHBwYEA6y
9CgkICQgHBwCGBQQAEMBAQEBAwMDBQUGBwCHCAkICQoCeAoJCAkIBwCHBgUEBAMDAQEBAQMDAwU
FBgCHBwgJCAkKvQqEBgUHBwCICQkJCgoKCwsLCwoKCgkJCQgHBwCfBgQBBYVRuh0FBQkIBwUFAgE
BAgUFBwgJCgkJCACGBAMBAQMEBgICQEIxfX39LwLrMwQFCAGHBQUCAQECEBQUHCAgJCQkIBwUEAwE
BAwQFBwgJIgICAwQFBQYGBwgICAgJCf0pCQkICAgIBWYGBUEAwICAgIDBAUFBgYHCAgICQkJatc
JCQkICAgHBgYFBQQDAgIKCQkICAgHBgYFBAQDAgICAgMEBAUGBgICAgJCQAFAAAAAAXA/QABwA
PABCAOACdAAABHwIjPwIDMzcZfZMDIYcVITUzESERJQcVDwYvBz8HHwYnKwEPDhEfDjMhMz8OES8
OKwEvDisBDw0B/wQKK3MmBQ6dMyeHKDWC090B1l79jQFoAQMEBgHCQkJCQgGBgQDAQEDBAYGCAk
JCQkHBwYEA6y9CgkICQgHBwCGBQQAEMBAQEBAwMDBQUGBwCHCAkICQoCeAoJCAkIBwCHBgUEBAM
DAQEBAQMDAwUFBgCHBwgJCAkKvQqEBgUHBwCICQkJCgoKCwsLCwoKCgkJCQgHBwCfBgQCFREigG4
SM/6wd3cBe/t9ff0vAtEzBAUICAcFBQIBAQIFBQcICAgJCQgHBgQDAQEDBAUHCAkiAgIDBAUFBgY
HCAgICQkJ/SkJCQkICAgHBgYFBQQDAgICAgMEBQUGBgICAgJCQkC1wkJCQgICAcGBgUFBAMCAgo
JCQgICAcGBgUEBAMCAgICAwQEBQYGBwgICAgJAAAAABIA3gABAAAAAEEEEAAAAABAAAAABAA8
AAQABAAAAAACAAcAEAAABAAAAAADAA8AFwABAAAAAAEAA8AJgABAAAAAAFAAAsANQABAAAAAA
GAA8AQAAABAAAAAAKCAwATwABAAAAAALABIAewADAAEECQAAAAIAjQADAAEECQABAB4AjwADAAE
ECQACAA4ArQADAAEECQADAB4AuWADAAEECQAEAB4A2QADAAEECQAFABYA9wADAAEECQAGAB4BDQA
DAAEECQAKAFgBKwADAAEECQALACQBgyBDb250ZXh0TWVudSAoMilSZWdlbGfGfYQ29udGV4dE1lbnU
gKDIpQ29udGV4dE1lbnUgKDIpVmVyc2lubiAxLjBDb250ZXh0TWVudSAoMilGb250IGdlbmVyYXR
lZCBlc2luZyBTeW5jZnVzaW9uIE1ldHJvIFN0dWRpb3d3dy5zeW5jZnVzaW9uLmNvbQAgAEMAbwB
uAHQAZQB4AHQATQB1AG4AdQAgACgAMgApAFIAZQBnAHUAbABhAHIAQwBvAG4AdAB1AHgAdABNAGU
AbgB1ACAaAAyACKAQwBvAG4AdAB1AHgAdABNAGUAbgB1ACAaAAyACKAVgB1AHIAcWBPAG8AbgA
gADEALgAwAEMAbwBuAHQAZQB4AHQATQB1AG4AdQAgACgAMgApAEYAbwBuAHQATQBnAGUAbgB1AHIA
YQBOAGUAZAAgAHUAcWBPAG4AZwAgAFMAeQBwAGMAZgB1AHMAaQBVAG4AIABNAGUAdABYAG8AIAAB
TAHQAdQBkAGkAbwB3AHcAdwAuAHMAeQBwAGMAZgB1AHMAaQBVAG4ALgBjAG8AbQAAAAACAAAAAA
AAAoAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAMBAGEDAQQAD01UX1Bhc3RlU3B1Y2lhbAxBXNVF9QYXN
0ZVRleHQAAA==) format('trueType');
font-weight: normal;
font-style: normal;
}
/* csslint ignore:stop */
.e-ddb-icons {
font-family: 'e-dropdown-btn';
font-style: normal;
font-variant: normal;
font-weight: normal;
line-height: 1;
text-transform: none;
}
@font-face {
font-family: 'ddb-icons';
src:
url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMj0gSRkAAAEoAAAAVmNtYXN0eDkAAABlAAAAADxnbHlmlh3
3NQAAAdwAAAJMaGVhZBKOK9sAAADQAAAAANmhoZWEHeANwAAAArAAAACRobXR4E6AAAAAAAYAAAA
UbG9yYQGOAegAAAHQAAAAADG1heHABEWBlAAABCAAAACBuYw1l1LBM9QAABCGAAAI9cG9zdmJntbU
AAAZoAAAAUAABAAADUv9qAFoEAAAAAADYgABAAAAAABAAAAAABAAAAABQABAAAAAQAAojXaQl8
PPPUACwPoAAAAANfSc4gAAAAA19JziAAA//oDyGPsAAAAACAACAAAAAEEEEAAAFaFkABAAAAAA
AAgAAAAoACgAAAP8AAAAAQAQPTAZAABQAAANoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnAwNS/2oAWgPsAJYAAAABAAAAAABAAAAAPoAAA
D6AAAA+gAAAPoAAAAAACAAAAAwAAABQAAwABAAAAFAAEACgAAAAEAAQAAQAA5wP//wAA5wD//wA
AAAEABAAAAAAEAAGADAAQAAAAAAAI4AwgEAAASYAAAwA//oDNQPsAA4AHQBIAAAALHgEOAScmJy4BNz4
BMzIFFqYHBgcGLqE2NzYzMyYBhgEXDqEHDqEHDqIWFxYXFjY3NjQ3PgE3HgEXFhQXhGqE3PgE3PgE

```

```

uAScuAScuASc+ATc+AQcLASYWAVEfFxo6IBkNCQIHcy8bCQG9BwIJDRkgOhoXHwoKgi/+TR1RDyE
OIxo+ExckFAQMfikwVhcMBwYlFRYkBwcMF1YwFCALDAQUIxcUPhojDiAOUR4cAQvEwwsB6gtDTyc
JCBsSKxYhJ0gWKxIaCQknUEILAYcCf2TPI0w2HBUMDg0sOzsaKQ4ONzcnIyYXNBgYNBcmIyc3OA8
GHRQaOzssDQ4mFRw2TiLOZGdBA/5vAZEDQQAEEAAAAA0qA+kABQANABcAHwAAARUzFSErAyERIZU
jNSEBIREhESMVITUjMyMVITUjNSMC733+iT8B9D4+/oj+igE4AXc//c4++j8BOT+7AbZ8+gF2/ks
Bdz4//ksB9AF2fHw+Pj8AAAIAAAAA7cD6QACACQAAAEhEwMOAQcVITUmJyY1ND8BIRcWFxYVFAC
GKwEVITUmJyYnASMCKP8AguQrOy0BGkIRHREkASstEgEEDhQxEQGaJxUcLP7PDAFNAVL+PHBHCBS
bBgsUKR8wX3owBg4NFgsQGxsDFx1zAyMAAAACAAAAAPKA+oAagATAAABFxEBDgEHHgEXETMRMxE
zETM1IQL+zPlabpADA5t0f2F+XP41afbMAZgBJwmYcHSbA/48A2r8lgNqfgAAAAASAN4AAQAAAAA
AAAAABAAAAQAAAAAAQAJAAEAQAAAAAAgAHAAoAAQAAAAAAAwAJABEAQAAAAABAAJABoAAQA
AAAAABQALACMAAQAAAAABgAJAC4AAQAAAAAACgAsADcAAQAAAAAACwASAGMAAwABBAkAAAACAHU
AAwABBAkAAQASAHcAAwABBAkAAgAOAIkAAwABBAkAAwASAJcAAwABBAkABAASAKkAAwABBAkABQA
WALsAAwABBAkABgASANEAAwABBAkACgBYAOMAawABBAkACwAkATsgZGRiLWljB25zUmVndWxhcmR
kYilpY29uc2RkYilpY29uc1ZlcnNpb24gMS4wZGRiLWljB25zRm9udCBnZW5lcmF0ZWQgdXNpbmc
gU3luY2Zlc2lvbiBNZXRYbyBTdHVkaW93d3cuc3luY2Zlc2lvbi5jb20AIABkAGQAYgAtAGkAYwB
vAG4AcwBSAGUAZwB1AGwAYQByAGQAZABiAC0AaQBjAG8AbgBzAGQAZABiAC0AaQBjAG8AbgBzAFY
AZQByAHMAaQBvAG4AIAAxAC4AMABkAGQAYgAtAGkAYwBvAG4AcwBGAG8AbgB0ACAAZwB1AG4AZQB
yAGEAdABLAGQAIAB1AHMAaQBvAGcAIABTAHkAbgBjAGYAdQBzAGkAbwBuACAATQBlAHQAcgBvACA
AUwB0AHUAZABpAG8AdwB3AHcALgBzAHkAbgBjAGYAdQBzAGkAbwBuAC4AYwBvAG0AAAAAAgAAAA
AAAAKAAAAAAAAAAAAAAAAAAAAAAAAAAAAFAQIBAwEEAQUBBgADY3V0CHBhc3RlXzAxBGZvbnc
OcGFyYS1tYXJrLS0tMDMAAA==) format('trueType');
font-weight: normal;
font-style: normal;
}
.e-db-icons {
  font-family: 'ddb-icons' !important;
  speak: none;
  font-size: 55px;
  font-style: normal;
  font-weight: normal;
  font-variant: normal;
  text-transform: none;
  line-height: 1;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}
#container {
  visibility: hidden;
}
#loader {
  color: #008cff;
  height: 40px;
  left: 45%;
  position: absolute;
  top: 45%;
  width: 30%;
}
.e-edit::before {
  content: '\ea9a';
}
.e-cut::before {
  content: '\e700';
}
.e-copy::before {
  content: '\e70a';
}
.e-paste::before {

```

```
content: '\e701';
}
.e-font::before {
content: '\e702';
}
.e-paragraph::before {
content: '\e703';
}
}
button {
margin: 25px 5px 20px 20px;
}
```

See Also

- [Integration with ListView component](#)
- [How to dynamically update option items in DropDownButton](#)

Accessibility in ##Platform_Name## Drop down button control

The Drop down button component followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Drop down button component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support | |

| [Section 508](#) Support | |

| Screen Reader Support | |

| Right-To-Left Support | |

| Color Contrast | |

| Mobile Device Support | |

| Keyboard Navigation Support | |

| [Accessibility Checker](#) Validation | |

| [Axe-core](#) Accessibility Validation | |

<style>

```
.post .post-content img {  
display: inline-block;  
margin: 0.5em 0;  
}  
</style>  
  
<div> - All  
features of the component meet the requirement.</div>  
  
<div> - Some features of the component do not meet the requirement.</div>  
  
<div> - The  
component does not meet the requirement.</div>
```

WAI-ARIA attributes

The Drop down button component followed the [WAI-ARIA] patterns to meet the accessibility. The following ARIA attributes are used in the Drop down button component:

| Attributes | Purpose |

| --- | --- |

| **role** | Indicates the Drop down button component as **button**, Drop down button popup as **menu**, and the dropdown popup action items as **menuitem**. |

| **aria-haspopup** | Indicates the availability of the popup element. |

| **aria-expanded** | Indicates whether the popup can be expanded or collapsed, as well as indicates whether its current state is expanded or collapsed. |

| **aria-owns** | Identifies an elements in order to define a visual, functional, or contextual parent/child relationship between DOM elements where the DOM hierarchy cannot be used to represent the relationship. |

| **aria-disabled** | Indicates that the element is perceivable but disabled, so it is not editable or otherwise operable. |

Keyboard interaction

The Dropdown button component followed the [keyboard interaction] guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Drop down button component.

| **Press** | **To do this** |

| --- | --- |

| **Esc** | Closes the popup. |

| **Enter** | Opens the popup, or activates the highlighted item and closes the popup. |

| **Space** | Opens the popup. |

| **Up** | Navigates up or to the previous action item. |

| Alt + Up Arrow | Closes the popup. |

| Alt + Down Arrow | Opens the popup. |

Ensuring accessibility

The Drop down component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Drop down button component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Drop down button component with accessibility tools.

See also

- [Accessibility in Syncfusion ##Platform_Name## components](#)

How To

Change caret icon in ##Platform_Name## Drop down button control

Dropdown arrow can be customized on popup open and close. It can be handled in [beforeOpen](#) and [beforeClose](#) event.

In the following example, the up arrow is updated on popup close and down arrow is updated on popup open using [beforeOpen](#) and [beforeClose](#) event by adding and removing [e-caret-up](#) class.

INDEX.JS

```
ej.base.enableRipple(true);
var items = [
  {
    text: 'Cut'
  },
  {
    text: 'Copy'
  },
  {
    text: 'Paste'
  }
];
var options = {
  items: items,
  beforeClose: function() {
    drpDownBtn.cssClass = '';
  },
  beforeOpen: function() {
    drpDownBtn.cssClass = 'e-caret-up';
  }
};
var drpDownBtn = new ej.splitbuttons.DropDownButton(options);
drpDownBtn.appendTo('#arrow');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 DropDownButton</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```

<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="arrow">Clipboard</button>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.e-caret {
    transform: rotate(0deg);
    transition: transform 200ms ease-in-out;
}
.e-caret-up .e-caret {
    transform: rotate(180deg);
}
button {
    margin: 25px 5px 20px 20px;
}

```

Create dropdownbutton with rounded corner in `##Platform_Name##` Drop down button control DropDownButton with rounded corner can be achieved by adding `border-radius` CSS property to button element.

In the following example, `e-round-corner` class is defined with `5px border-radius` property and added that class to button element using `cssClass` property.

INDEX.JS

```
ej.base.enableRipple(true);
//Initialize action items.
var items = [
    {
        text: 'Cut'
    },
    {
        text: 'Copy'
    },
    {
        text: 'Paste'
    }
];
// initialize DropDownButton component
var drpDownBtn = new ej.splitbuttons.DropDownButton({items: items, cssClass:
'e-round-corner'});
// Render initialized DropDownButton.
drpDownBtn.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 DropDownButton</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="styles.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="element">Clipboard</button>
    </div>
```

```
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

STYLES.CSS

```
#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.e-round-corner {
    border-radius: 5px;
}
```

Create right to left dropdownbutton in ##Platform_Name## Drop down button control
DropDownButton component has RTL support. This can be achieved by setting [enableRtl](#) as true.

The following example illustrates how to enable right-to-left support in DropDownButton component.

INDEX.JS

```
ej.base.enableRipple(true);
var items = [
    {
        text: 'Edit'
    },
    {
        text: 'Delete'
    },
    {
        text: 'Mark as Read'
    },
    {
        text: 'Like Message'
    }
];
//To position the icon to the left of the text on a DropDownButton.
var drpDownBtn = new ej.splitbuttons.DropDownButton({iconCss: 'ddb-icons e-message', items: items, enableRtl: true}, '#iconbutton');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 DropDownButton</title>
```

```

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="styles.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="iconbutton">Message</button>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
@font-face {
font-family: 'e-db-icons';
src:
url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAIAIAAwAgTlMvMj0jSRoAAAEoAAAAVmNtYXDnFudgAAABkAAAApnbHlmSrK
TCAAAAdgAAAC4aGVhZBKtK8cAAADQAAAAANmhoZWEHmQNtAAAArAAAACRobXR4D7gAAAAAYAAAAA
QbG9jYQB4ADoAAAHMAAAACm1heHABEAAAYAAABCAAAACBuYW1lH00mDAAAAPAAAAJJcG9zdIwSr0
AAATcAAAATQABAAADUv9qAFoEAAAA//4D6gABAAAAAAAAAAAAAAAAABAABAAAAQAAGc/PS18
PPPUACwPoAAAAANfSc3wAAAAA19JzfAAAAAAD6gPqAAACAACAAAAAAAAAAAAEAAAwAAgAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQPuAZAABQAAANoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wPnBQNS/2oAWgPqAJYAAAAABAAAAAAAAABAAAAAPoAAA
D6AAAA+gAAAAAAAAIAAADAAAAFAADAAEAAAAUAAQAJgAAAAQABAABADnBf//AADnA//AAAAQA
EAAAAAQACAAMAAAAAAAAAHAA6AFwAAAAACAAAAAPqA2UABgAKAAA3IREjCQEjBRcBIQID6AL+Dv4
NAQEY3QG4/I+IAsL+GAHonroBcwAAAAIAAAAAA8YD6gAFAAoAADchESMJASUHCQImA6AD/jL+MQE
EywGWAZb+agICX/4+AcLXsv6cAWQBZAAAAEAAAAAAAA+oD6gALAAATCQEXCQE3CQEnCQECATP+zcI
BMgEzwf7OATLB/s3+zgMp/s3+zsIBM/7NwgEyATPB/s4BMgAAAAASAN4AAQAAAAAAAAABAAAAQA
AAAAAQAKAAEAAQAAAAAAAAgAHAAsAAQAAAAAAAAAwAKABIAAQAAAAAABAABAAQAAAAAABQALACY

```

```
AAQAAAAAABgAKADEAAQAAAAAACgAsADsAAQAAAAAACwASAGCAAwABBAkAAAAACAHAwABBAkAAQA
UAHsAAwABBAkAAgAOAI8AAwABBAkAAwAUAJ0AAwABBAkABAAUALEAAwABBAkABQAWAMUAAwABBAk
ABgAUANsAAwABBAkACgBYAO8AAwABBAkACwAkAUcgZS1kYi1pY29uc1JlZ3VsYXJlLWRiLWlj25
zZS1kYi1pY29uc1ZlcnNpb24gMS4wZS1kYi1pY29uc0ZvbnQgZ2VuZXJhdGVkIHVzaW5nIFN5bmN
mdXNpb24gTWV0cm8gU3RlZGlvd3d3LnN5bmNmdXNpb24uY29tACAAZQAtAGQAYgAtAGkAYwBvAG4
AcwBSAGUAZwBLAGwAYQByAGUALQBkAGIALQBpAGMAbwBuAHMAZQAtAGQAYgAtAGkAYwBvAG4AcwB
WAGUAcgBzAGkAbwBuACAAMQAUADAAZQAtAGQAYgAtAGkAYwBvAG4AcwBGAG8AbgB0ACAAZwBLAG4
AZQByAGEAdABLAGQAIAB1AHMAaQBuAGcAIABTAHkAbgBjAGYAdQBzAGkAbwBuACAATQB1AHQAcgB
vACAAUwB0AHUAZABpAG8AdwB3AHcALgBzAHkAbgBjAGYAdQBzAGkAbwBuAC4AYwBvAG0AAAAAAGA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAEAQIBAwEEAQUADG1lc3NhZ2UtZW50bWVpATyZWf
kLXVucmVhZAZkZWxldGUAAAAAAAA==) format('trueType');
font-weight: normal;
font-style: normal;
}
.ddb-icons {
  font-family: 'e-db-icons' !important;
speak: none;
font-size: 55px;
font-style: normal;
font-weight: normal;
font-variant: normal;
text-transform: none;
line-height: 1;
-webkit-font-smoothing: antialiased;
-moz-osx-font-smoothing: grayscale;
}
.e-message::before {
  content: '\e703';
}
#loader {
  color: #008cff;
  height: 40px;
  left: 45%;
  position: absolute;
  top: 45%;
  width: 30%;
}
button {
  margin: 25px 5px 20px 20px;
}
```

Customize icon and width in ##Platform Name## Drop down button control

Width of the DropDownButton can be customized by setting required width to the dropdown element.

The following UI can be achieved by setting `iconPosition` as `Top`, width as `85px` and size of the font icon as `40px` by adding `e-custom` class.

INDEX.JS

```
ej.base.enableRipple(true);
var items = [
  {
    text: 'Find'
  },
  {
    text: 'Replace'
  }
];
```

```

    },
    {
        text: 'Go To'
    },
    {
        text: 'Go To Special'
    }
  ]];
// To initialize DropDownButton with `e-custom` class.
var drpDownBtn = new ej.splitbuttons.DropDownButton({
  iconCss: 'e-icons e-search',
  cssClass: 'e-custom',
  items: items,
  iconPosition: 'Top'
}, '#iconbutton'
);

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 DropDownButton</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <button id="iconbutton">Find &#38; Select</button>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```
#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.e-search::before {
    content: '\e993';
}
button {
    margin: 25px 5px 20px 20px;
}
.e-dropdown-btn.e-custom {
    width: 85px;
}
.e-dropdown-btn.e-custom .e-search::before {
    font-size: 40px;
}
```

Disable a dropdownbutton in ##Platform_Name## Drop down button control

DropDownButton component can be enabled/disabled by giving [disabled](#) property. It can be disabled by setting disabled property as true.

INDEX.JS

```
ej.base.enableRipple(true);
var items = [
    {
        text: 'Edit'
    },
    {
        text: 'Delete'
    },
    {
        text: 'Mark as Read'
    },
    {
        text: 'Like Message'
    }
];
//To position the icon to the left of the text on a DropDownButton.
var drpDownBtn = new ej.splitbuttons.DropDownButton({iconCss: 'ddb-icons e-message', items: items, disabled: true}, '#iconbutton');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 DropDownButton</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
```



```

<meta name="author" content="Syncfusion">
<link href="styles.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="iconbutton">Message</button>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
@font-face {
    font-family: 'e-db-icons';
    src:
url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMj0jSRoAAAEoAAAAVmNtYXDNFudgAAABkAAAAADpnbHlmSrK
TCAAAAdgAAAC4aGVhZBktK8cAAADQAAAAANmhoZWEHmQNtAAAArAAAACRobXR4D7gAAAAAYAAAAA
QbG9jYQB4ADoAAAHMAAAACm1heHABEAAAYAAABCAAAACBuYW1lH00mDAAAAPAAAAJJCg9zdIwSr0
AAATcAAAATQABAAADUv9qAFoEAAAA//4D6gABAAAAAAAAAAAAAAAAAAAAAAAAAABAAAAAAQAAgc/PS18
PPPUACwPoAAAAANfSc3wAAAAA19JzfAAAAAAD6gPqAAAAACAACAAAAAAAAAAAAEAAAAEAAwAAgAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAAAQPUAZAABQAAAAnoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wPnBQNS/2oAWgPqAJYAAAABAAAAAAAAABAAAAAPoAAA
D6AAAA+gAAAAAAAAIAAADAAAAFAADAAEAAAAUAAQAjgAAAAQABAABAADnBf//AADnA///AAAAQA
EAAAAQAQCAAAAAAAAAAAAAHAA6AFwAAAACAAAAAAPqA2UABgAKAAA3IREjCQEjBRcBIQID6AL+Dv4
NAQEY3QG4/I+IAsL+GAHonroBcwAAAAIAAAAAA8YD6gAFAAoAADchESMJASUHCQImA6AD/jL+MQE
EywGWAZb+agICX/4+AcLXsv6cAWQBZAAAAAEAAAAA+oD6gALAAATCQEXCQE3CQEnCQECATP+zcI
BMgEzwf7OATLB/s3+zsMp/s3+zsIBM/7NwgEyATPB/s4BMgAAAAASAN4AAQAAAAAAAAABAAAAQA
AAAAAQAKAAEAAQAAAAAAAgAHAAsAAQAAAAAAAwAKABIAAQAAAAABAAKABWAAQAAAAABQALACY
AAQAAAAABgAKADEAAQAAAAAACgAsADsAAQAAAAAACwASAGCAAwABBAKAAAACAHkAAwABBAKAAQA
UAHsAAwABBAKAAgAOAI8AAwABBAKAAwAUAJ0AAwABBAKABAAUALEAAwABBAKABQAWAMUAwABBAK
ABgAUANsAAwABBAKACgBYAO8AAwABBAKACwAkAUcgZS1kYi1pY29uc1JlZ3VsYXJlLWwRiLWljb25

```

```

zZS1kYi1pY29uc1ZlcnNpb24gMS4wZS1kYi1pY29uc0ZvbnQgZ2VuZXJhdGVkIHVzaW5nIFN5bmN
mdXNpb24gTWV0cm8gU3R1ZGlvd3d3LnN5bmNmdXNpb24uY29tACAAZQAtAGQAYgAtAGkAYwBvAG4
AcwBSAGUAZwB1AGwAYQByAGUALQBkAGIALQBpAGMABwBuAHMAZQAtAGQAYgAtAGkAYwBvAG4AcwB
WAGUAcgBzAGkAbwBuACAAMQAuADAAZQAtAGQAYgAtAGkAYwBvAG4AcwBGAG8AbgB0ACAAZwB1AG4
AZQByAGEAdAB1AGQAIAB1AHMAaQBwAGcAIABTAHkAbgBjAGYAdQBzAGkAbwBuACAATQB1AHQAcgB
vACAAUwB0AHUAZABpAG8AdwB3AHcALgBzAHkAbgBjAGYAdQBzAGkAbwBuAC4AYwBvAG0AAAAAAGa
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAEAQIBAwEEAQUADG11c3NhZ2UtZW50bWVpAtyZWF
kLXVucmVhZAZkZWxldGUAAAAA==) format('true');
font-weight: normal;
font-style: normal;
}
.ddb-icons {
  font-family: 'e-db-icons' !important;
  speak: none;
  font-size: 55px;
  font-style: normal;
  font-weight: normal;
  font-variant: normal;
  text-transform: none;
  line-height: 1;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}
.e-message::before {
  content: '\e703';
}
#loader {
  color: #008cff;
  height: 40px;
  left: 45%;
  position: absolute;
  top: 45%;
  width: 30%;
}
button {
  margin: 25px 5px 20px 20px;
}

```

Group popup items with listview component in `##Platform_Name##` Drop down button control Header in popup items is possible in DropDownButton by templating entire popup with ListView. Create ListView with id `#listview` and provide it as a [target](#) for DropDownButton.

In the following example, ListView element is given as `target` to DropDownButton and header can be achieved by [groupBy](#) property.

INDEX.JS

```

ej.base.enableRipple(true);
var ddbOption = {
  target: '#listview',
  iconCss: 'e-icons e-down',
  cssClass: 'e-caret-hide'
};
var drpDownBtn = new ej.splitbuttons.DropDownButton(ddbOption);
drpDownBtn.appendTo('#element');
var dataSource = [

```

```

    { class: 'data', text: 'Print', id: 'data1', category: 'Customize Quick
Access Toolbar' },
    { class: 'data', text: 'Save As', id: 'data2', category: 'Customize
Quick Access Toolbar' },
    { class: 'data', text: 'Update Folder', id: 'data3', category:
'Customize Quick Access Toolbar' },
    { class: 'data', text: 'Reply', id: 'data4', category: 'Customize Quick
Access Toolbar' },
    { class: 'data', text: 'Reply All', id: 'data4', category: 'Customize
Quick Access Toolbar' },
    { class: 'data', text: 'Forward', id: 'data4', category: 'Customize
Quick Access Toolbar' },
    { class: 'data', text: 'Delete', id: 'data4', category: 'Customize Quick
Access Toolbar' },
    { class: 'data', text: 'Undo', id: 'data4', category: 'Customize Quick
Access Toolbar' },
];
//Initialize ListView component
var listViewInstance = new ej.lists.ListView({
    dataSource: dataSource,
    //map the appropriate columns to fields property
    fields: { text: 'text', groupBy: 'category' },
    showCheckBox: true
});
//Render initialized ListView
listviewInstance.appendTo("#listview");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 DropDownButton</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">

```

```

        <button id="element"></button>
        <div id="listview"></div>
    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.e-down::before {
    content: '\e969';
}
#listview {
    display: block;
    max-width: 600px;
    margin: auto;
    border: 1px solid #dddddd;
    border-radius: 3px;
}

```

Hide dropdown arrow in ##Platform_Name## Drop down button control

You can hide the dropdown arrow from the DropDownButton by adding class `e-caret-hide` to DropDownButton element using [cssClass](#) property.

INDEX.JS

```

ej.base.enableRipple(true);
var items = [
    {
        text: 'Cut'
    },
    {
        text: 'Copy'
    },
    {
        text: 'Paste'
    }
];
var drpDownBtn = new ej.splitbuttons.DropDownButton({items: items});
drpDownBtn.appendTo('#hide');

```

```
drpDownBtn.element.classList.add('e-caret-hide');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 DropDownButton</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <button id="hide">Clipboard</button>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

STYLES.CSS

```
#container {
  visibility: hidden;
}
#loader {
  color: #008cff;
  height: 40px;
  left: 45%;
  position: absolute;
  top: 45%;
  width: 30%;
}
```

Open a dialog on popup item click in `##Platform_Name##` Drop down button control

This section explains about how to open a dialog on DropDownButton popup item click. This can be achieved by handling dialog open in [select](#) event of the DropDownButton.

In the following example, Dialog will open while selecting **Other Folder...** item.

INDEX.JS

```
ej.base.enableRipple(true);
var dialog = new ej.popups.Dialog({
  content: "Move Items To `Web Team`,",
  header: 'Move Items',
  buttons: [{
    buttonModel: {
      isPrimary: true,
      content: 'OK',
      cssClass: 'e-flat'
    },
    click: function () {
      this.hide();
    }
  }],
  width: '250px',
  visible: false,
  position: {X: 100, Y: 100}
});
dialog.appendTo('#dialog');
var items = [
  {
    text: 'Archive'
  },
  {
    text: 'Inbox'
  },
  {
    text: 'HR Portal'
  },
  {
    separator: true
  },
  {
    text: 'Other Folder...'
  },
  {
    text: 'Copy to Folder'
  }
];
var ddbOption = {
  iconCss: 'ddb-icons e-folder',
  cssClass: 'e-vertical',
  items: items,
  iconPosition: 'Top',
  select: select
};
//To position the icon to the left of the text on a DropDownButton.
var drpDownBtn = new ej.splitbuttons.DropDownButton(ddbOption,
'#iconbutton');
function select (args) {
```

```

    if (args.item.text === 'Other Folder...') {
        dialog.show();
    }
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Button</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="dialog"></div>
    <button id="iconbutton">Move</button>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
  visibility: hidden;
}
#loader {
  color: #008cff;
  height: 40px;
  left: 45%;
  position: absolute;
  top: 45%;
  width: 30%;
}

```

```

}
@font-face {
font-family: 'e-db-icons';
src:
url (data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMj0jSRoAAAEoAAAAVmNtYXDNFudgAAABkAAAAADpnbHlmSrK
TCAAAAdgAAAC4aGVhZBKtK8cAAADQAAAAANmhoZWEHmQNtAAAArAAAACRobXR4D7gAAAAAYAAAA
QbG9jYQB4ADoAAAHMAAAACm1heHABEAAAYAAABCAAAACBuYW1lH00mDAAAAPAAAAJcG9zdIwksr0
AAATcAAAATQABAAADUv9qAFoEAAAA//4D6gABAAAAAAAAAAAAAAAAABABAAAAQAAGc/PS18
PPPUACwPoAAAAANfSc3wAAAAA19JzfAAAAAAD6gPqAAAAACAACAAAAAAAAAAAAEAAAwAAgAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAAPuAZAABQAAAAnoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wPnBQNS/2oAWgPqAJYAAAAABAAAAAAAAABAAAAAPoAAA
D6AAAA+gAAAAAAIAAAADAAAAFAADAAEAAAAUAAQAjgAAAAQABAABADnBf//AADnA///AAAAQA
EAAAAQACAAMAAAAAAAAAHAA6AFwAAAAACAAAAAPqA2UABgAKAAA3IREjCQEjBRcBIQID6AL+Dv4
NAQEY3QG4/I+IAsL+GAHonroBcwAAAAIAAAAAA8YD6gAFAAoAADchESMJASUHCQImA6AD/jL+MQE
EywGWAZb+agICX/4+AcLXsv6cAWQBZAAAAEAAAAAAAA+oD6gALAAATCQEXCQE3CQEnCQECATP+zcI
BMgEzwf7OATLB/s3+zgMp/s3+zsIBM/7NwgEyATPB/s4BMgAAAAASAN4AAQAAAAAAAAABAAAAQA
AAAAAQAKAAEAAQAAAAAAAAgAHAAsAAQAAAAAAAwAKABIAAQAAAAABAAKABwAAQAAAAABQALACY
AAQAAAAABgAKADEAAQAAAAACgAsADsAAQAAAAACwASAGCAAwABBAkAAAACAHkAAwABBAkAAQA
UAHsAAwABBAkAAgAOAI8AAwABBAkAAwAUAJ0AAwABBAkABAAUALEAAwABBAkABQAWAMUAwABBAk
ABgAUANsAAwABBAkACgBYAO8AAwABBAkACwAkAUcgZS1kYi1pY29uc1JlZ3VsYXJlLWw1LWljZj25
zZS1kYi1pY29uc1ZlcnNpb24gMS4wZS1kYi1pY29uc0ZvbnQgZ2VuZXJhdGVkIHVzaW5nIFN5bmN
mdXNpb24gTWV0cm8gU3R1ZG1vd3d3LnN5bmNmdXNpb24uY29tACAAZQAtAGQAYgAtAGkAYwBvAG4
AcwBSAGUAZwB1AGwAYQByAGUALQBkAGIALQBpAGMABwBuAHMAZQAtAGQAYgAtAGkAYwBvAG4AcwB
WAGUAcgBzAGkAbwBuACAAMQAuADAAZQAtAGQAYgAtAGkAYwBvAG4AcwBGAG8AbgB0ACAAZwB1AG4
AZQByAGEAdABlAGQAIABlAHMAaQBuAGcAIABTAHkAbgBjAGYAdQBzAGkAbwBuACAATQBlAHQAcgB
vACAAUwB0AHUAZABpAG8AdwB3AHcALgBzAHkAbgBjAGYAdQBzAGkAbwBuAC4AYwBvAG0AAAAAAgA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAEAEQIBAwEEAQUADG1lc3NhZ2UtZWVpZAtyZWV
kLXVucmVhZAZkZWxldGUAAAAAAAA==) format('trueType');
font-weight: normal;
font-style: normal;
}
.ddb-icons {
font-family: 'e-db-icons' !important;
speak: none;
font-size: 55px;
font-style: normal;
font-weight: normal;
font-variant: normal;
text-transform: none;
line-height: 1;
-webkit-font-smoothing: antialiased;
-moz-osx-font-smoothing: grayscale;
}
.e-folder::before {
content: '\e703';
}

```

Position popup open in ##Platform_Name## Drop down button control

Popup open position can be changed according to the requirement. Popup open position can be changed in [open](#) event by setting **top** and **left** for the popup element.

In the following example, the **top** position of the popup element is changed in **open** event.

INDEX.JS


```

ej.base.enableRipple(true);
//Initialize action items.
var items = [
    {
        text: 'Cut'
    },
    {
        text: 'Copy'
    },
    {
        text: 'Paste'
    }
];
// initialize DropDownButton component
var drpDownBtn = new ej.splitbuttons.DropDownButton({
    items: items,
    cssClass: 'e-caret-up',
    open: onOpen
});
// Render initialized DropDownButton.
drpDownBtn.appendTo('#element');
function onOpen(args) {
    args.element.parentElement.style.top =
drpDownBtn.element.getBoundingClientRect().top -
args.element.parentElement.offsetHeight + 'px';
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 DropDownButton</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="styles.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="element">Clipboard</button>
    </div>
</script>

```

```

var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
button {
    margin: 25% 5px 20px 30%;
}
.e-caret-up .e-caret::before {
    content: '\e918';
}

```

Underline a character in the item text in ##Platform_Name## Drop down button control

Underline a particular character in a text can be handled in [beforeItemRender](#) event by adding `<u>` tag in between the text and given as innerHTML in `li` rendering.

In the following example, `C` is underlined in the text `Copy`.

INDEX.JS

```

ej.base.enableRipple(true);
var items = [
    {
        text: 'Cut'
    },
    {
        text: 'Copy'
    },
    {
        text: 'Paste'
    }
];
var ddbOption = {
    items: items,
    beforeItemRender: itemRender
}
var drpDownBtn = new ej.splitbuttons.DropDownButton(ddbOption);
drpDownBtn.appendTo('#element');
function itemRender(args) {
    if (args.item.text === 'Copy') {

```

```
        args.element.innerHTML = '<u>C</u>opy';  
    }  
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>  
  <title>EJ2 DropDownButton</title>  
  <meta charset="utf-8">  
  <meta name="viewport" content="width=device-width, initial-scale=1.0">  
  <meta name="description" content="Typescript UI Controls">  
  <meta name="author" content="Syncfusion">  
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">  
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">  
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">  
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">  
  <link href="styles.css" rel="stylesheet">  
  
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js" type="text/javascript"></script>  
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type="text/javascript"></script>  
</head>  
<body>  
  
  <div id="container">  
    <button id="element">Clipboard</button>  
  </div>  
<script>  
var ele = document.getElementById('container');  
if(ele) {  
  ele.style.visibility = "visible";  
}  
</script>  
<script src="index.js" type="text/javascript"></script>  
</body></html>
```

STYLES.CSS

```
#container {  
  visibility: hidden;  
}  
#loader {  
  color: #008cff;  
  height: 40px;  
  left: 45%;  
  position: absolute;  
  top: 45%;  
  width: 30%;  
}
```

DropDownList

Tags in ##Platform_Name## Drop down list control

The DropDownList can be initialized on three different tags as described in below. Though it is initialized in different tags, the UI appearance and built-in features behave in the same way.

Select element

When a DropDownList is initialized on SELECT element, the list items can be assigned through the option tag of the HTML select element.

- The nested items are wrapped and grouped based on the tag that is available

within the `<select>` element, by default.

- You can preselect the option by setting the `selected` attribute to an option tag.

INDEX.JS

```
// initialize DropDownList component
let ddlObject = new ej.dropdowns.DropDownList({
  placeholder: "Select a vegetable",
  popupHeight: "200px"
});
// render initialized DropDownList
ddlObject.appendTo('#selectElement');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DropDownList</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:0 auto; width:250px;">
    <select id="selectElement">
```

```

        <optgroup label="Beans">
            <option value="1">Chickpea</option>
            <option value="2">Green bean</option>
            <option value="3">Horse gram</option>
        </optgroup>
        <optgroup label="Leafy and Salad">
            <option value="4" selected="selected">Cabbage</option>
            <option value="5">Spinach</option>
            <option value="6">Wheat grass</option>
        </optgroup>

    </select>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

UL element

The DropDownList can be initialized through `` element which contains a collection of `` element. The `` items act as a popup list items of the DropDownList. The inner text of the `` element is considered both as text and value fields.

INDEX.JS

```

//initiates the component
let ddlObject = new ej.dropdowns.DropDownList({
    placeholder: "Select a vegetable"
});
// render initialized DropDownList
ddlObject.appendTo('#ulElement');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 DropDownList</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>

```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin:0 auto; width:250px;">
        <ul id="ulElement">
            <li>Cabbage</li>
            <li>Spinach</li>
            <li>Wheat grass</li>
        </ul>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Input element

The DropDownList has also be rendered through `<input>` element with an array of either simple or complex data that is set through the [dataSource](#) property. It can retrieve data from local data sources as well as remote data services.

Detailed information about the data binding with an example is available in: [Data Binding to DropDownList](#)

Data binding in ##Platform_Name## Drop down list control

The DropDownList loads the data either from local data sources or remote data services using the [dataSource](#) property. It supports the data type of `array` or `DataManager`.

The DropDownList also supports different kinds of data services such as OData, OData V4, and Web API, and data formats such as XML, JSON, and JSONP with the help of `DataManager` adaptors.

Fields	Type	Description
text	string	Specifies the display text of each list item.
value	number or string	Specifies the hidden data value mapped to each list item that should contain a unique value.
groupBy	string	Specifies the category under which the list item has to be grouped.
iconCss	string	Specifies the icon class of each list item.

When binding complex data to the DropDownList, fields should be mapped correctly. Otherwise, the selected item remains undefined.

Binding local data

Local data can be represented in two ways as described below.

1. Array of simple data

The DropDownList has support to load array of primitive data such as strings and numbers. Here, both value and text field act the same.

INDEX.JS

```
var sportsData = ['Badminton', 'Cricket', 'Football', 'Golf', 'Tennis'];
// initialize DropDownList component
var listObj = new ej.dropdowns.DropDownList({
  dataSource: sportsData,
  // set placeholder to DropDownList input element
  placeholder: "Select games"});
listObj.appendTo('#ddlelement');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DropDownList</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:0 auto; width:250px;">
    <input type="text" tabindex="1" id="ddlelement">
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

2. Array of JSON data

The DropDownList can generate its list items through an array of complex data. For this, the appropriate columns should be mapped to the [fields](#) property.

In the following example, **Id** column and **Game** column from complex data have been mapped to the **value** field and **text** field, respectively.

INDEX.JS

```
let sportsData = [
  { Id: 'Game1', Game: 'Badminton' },
  { Id: 'Game2', Game: 'Basketball' },
  { Id: 'Game3', Game: 'Cricket' },
  { Id: 'Game4', Game: 'Football' },
  { Id: 'Game5', Game: 'Golf' },
  { Id: 'Game6', Game: 'Hockey' },
  { Id: 'Game7', Game: 'Rugby' },
  { Id: 'Game8', Game: 'Snooker' }
];

// initialize DropDownList component
var listObj = new ej.dropdowns.DropDownList({
  //set the data to dataSource property
  dataSource: sportsData,
  // maps the appropriate column to fields property
  fields: { value: 'Game' },
  // set placeholder to DropDownList input element
  placeholder: "Find a game"});
listObj.appendTo('#ddlelement');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DropDownList</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:0 auto; width:250px;">
    <br>
    <input type="text" id="ddlelement">
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
```



```

    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

3. Array of Complex data

The DropDownList can generate its list items through an array of complex data. For this, the appropriate columns should be mapped to the [fields](#) property.

In the following example, `Code.Id` column and `Country.Name` column from complex data have been mapped to the `value` field and `text` field, respectively.

INDEX.JS

```

let countriesData = [
    { Country: { Name: 'Australia' }, Code: { Id: 'AU' } },
    { Country: { Name: 'Bermuda' }, Code: { Id: 'BM' } },
    { Country: { Name: 'Canada' }, Code: { Id: 'CA' } },
    { Country: { Name: 'Cameroon' }, Code: { Id: 'CM' } },
    { Country: { Name: 'Denmark' }, Code: { Id: 'DK' } },
    { Country: { Name: 'France' }, Code: { Id: 'FR' } }
];
// initialize DropDownList component
var listObj = new ej.dropdowns.DropDownList({
    //set the data to dataSource property
    dataSource: countriesData,
    // maps the appropriate column to fields property
    fields: { value: 'Country.Name' },
    // set placeholder to DropDownList input element
    placeholder: "Find a country"});
listObj.appendTo('#ddlelement');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DropDownList</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>

```

```

<body>

    <div id="container" style="margin:0 auto; width:250px;">
        <br>
        <input type="text" id="ddlelement">
    </div>
</body>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Binding remote data

The DropDownList supports retrieval of data from remote data services with the help of **DataManager** component. The [Query](#) property is used to fetch data from the database and bind it to the DropDownList.

The following sample displays the first 6 contacts from “Customers” table of the **Northwind** Data Service.

INDEX.JS

```

//initiates the component
var ddlObject = new ej.dropdowns.DropDownList({
    //bind the data manager instance to dataSource property
    dataSource: new ej.data.DataManager({
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/',
        adaptor: new ej.data.ODataV4Adaptor(),
        crossDomain: true
    }),
    //bind the Query instance to query property
    query: new ej.data.Query().from('Employees').select(['FirstName',
    'City', 'EmployeeID']).take(6),
    //map the appropriate columns to fields property
    fields: { value: 'FirstName' },
    //set the placeholder to DropDownList input
    placeholder: "Select an employee",
    //sort the resulted items
    sortOrder: 'Ascending'
});
//render the component
ddlObject.appendTo('#ddlelement');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 DropDownList</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin:0 auto; width:250px;">
        <br>
        <input type="text" id="ddlelement">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [How to load data using template](#)
- [How to group the data using header](#)
- [How to filter the bound data](#)
- [How to get the count of the data when using remote data](#)
- [How to achieve cascading](#)
- [How to add item in between the options](#)
- [How to remove an item](#)
- [How to preselect the items in dropdownlist](#)

Templates in ##Platform_Name## Drop down list control

The DropDownList has been provided with several options to customize each list item, group title, selected value, header, and footer elements. It uses the Essential JS 2 [Template engine](#) to compile and render the elements properly.

Item template

The content of each list item within the DropDownList can be customized with the help of [itemTemplate](#) property.

In the following sample, each list item is split into two columns to display relevant data's.

INDEX.JS

```
//initiates the component
```

```

var ddlObject = new ej.dropdowns.DropDownList({
  //bind the data manager instance to dataSource property
  dataSource: new ej.data.DataManager({
    url: 'https://services.odata.org/V4/Northwind/Northwind.svc/',
    adaptor: new ej.data.ODataV4Adaptor(),
    crossDomain: true
  }),
  //bind the Query instance to query property
  query: new ej.data.Query().from('Employees').select(['FirstName',
'City', 'EmployeeID']).take(6),
  //map the appropriate columns to fields property
  fields: { value: 'FirstName' },
  //set the placeholder to DropDownList input
  placeholder: "Select an employee",
  //sort the resulted items
  sortOrder: 'Ascending',
  //set the value to itemTemplate property
  itemTemplate: "<span><span class='name'>${FirstName}</span><span class
='city'>${City}</span></span>"
});
//render the component
ddlObject.appendTo('#ddlelement');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DropDownList</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:0 auto; width:250px;">
    <input type="text" id="ddlelement">
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>

```

```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Value template

The currently selected value that is displayed by default on the DropDownList input element can be customized using the [valueTemplate](#) property.

In the following sample, the selected value is displayed as a combined text of both **FirstName** and **City** in the DropDownList input, which is separated by a hyphen.

INDEX.JS

```
//initiates the component
var ddlObject = new ej.dropdowns.DropDownList({
  //bind the data manager instance to dataSource property
  dataSource: new ej.data.DataManager({
    url: 'https://services.odata.org/V4/Northwind/Northwind.svc/',
    adaptor: new ej.data.ODataV4Adaptor(),
    crossDomain: true
  }),
  //bind the Query instance to query property
  query: new ej.data.Query().from('Employees').select(['FirstName',
'City', 'EmployeeID']).take(6),
  //map the appropriate columns to fields property
  fields: { value: 'FirstName' },
  //set the placeholder to DropDownList input
  placeholder: "Select an employee",
  //sort the resulted items
  sortOrder: 'Ascending',
  //set the value to itemTemplate property
  itemTemplate: "<span><span class='name'>${FirstName}</span><span class
='city'>${City}</span></span>",
  //set the value to valueTemplate property
  valueTemplate: "<span>${FirstName} - ${City}</span>"
});
//render the component
ddlObject.appendTo('#ddlelement');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DropDownList</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin:0 auto; width:250px;">
        <input type="text" id="ddlelement">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Group template

The group header title under which appropriate sub-items are categorized can also be customize with the help of [groupTemplate](#) property. This template is common for both inline and floating group header template.

In the following sample, employees are grouped according to their city.

INDEX.JS

```

var groupPredicate = new ej.data.Predicate('City',
'equal','london').or('City','equal','seattle');
//initiates the component
var ddlObject = new ej.dropdowns.DropDownList({
    //bind the data manager instance to dataSource property
    dataSource: new ej.data.DataManager({
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/',
        adaptor: new ej.data.ODataV4Adaptor(),
        crossDomain: true
    }),
    //bind the Query instance to query property
    query: new ej.data.Query().from('Employees').select(['FirstName',
'City','EmployeeID']).take(6).where(groupPredicate),
    //map the appropriate columns to fields property
    fields: { value: 'FirstName', groupBy: 'City'},
    //set the placeholder to DropDownList input
    placeholder:"Select an employee",
    //sort the resulted items
    sortOrder: 'Ascending',
    //set the value to groupTemplate
    groupTemplate: "<strong>${City}</strong>"
});
//render the component
ddlObject.appendTo('#ddlelement');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DropDownList</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:0 auto; width:250px;">
    <input type="text" id="ddlelement">
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Header template

The header element is shown statically at the top of the popup list items within the DropDownList, and any custom element can be placed as a header element using the [headerTemplate](#) property.

In the following sample, the list items and its headers are designed and displayed as two columns similar to multiple columns of the grid.

INDEX.JS

```

//initiates the component
var ddlObject = new ej.dropdowns.DropDownList({
  //bind the data manager instance to dataSource property
  dataSource: new ej.data.DataManager({
    url: 'https://services.odata.org/V4/Northwind/Northwind.svc/',
    adaptor: new ej.data.ODataV4Adaptor(),
    crossDomain: true
  }),
  //bind the Query instance to query property
  query: new ej.data.Query().from('Employees').select(['FirstName',
'City', 'EmployeeID']).take(6),
  //map the appropriate columns to fields property

```

```

fields: { value: 'FirstName'},
//set the placeholder to DropDownList input
placeholder:"Select an employee",
//sort the resulted items
sortOrder: 'Ascending',
//set the value to headerTemplate
headerTemplate: "<span class='head'><span class='name'>Name</span><span
class='city'>City</span></span>",
//set the value to item template
itemTemplate: "<span class='item' ><span
class='name'>${FirstName}</span><span class='city'>${City}</span></span>"
});
//render the component
ddlObject.appendTo('#ddlelement');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DropDownList</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:0 auto; width:250px;">
    <input type="text" id="ddlelement">
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Footer template

The DropDownList has options to show a footer element at the bottom of the list items in the popup list. Here, you can place any custom element as a footer element using the [footerTemplate](#) property.

In the following sample, footer element displays the total number of list items present in the DropDownList.

INDEX.JS

```
var sportsData = [ "Basketball", "Cricket", "Football", "Golf"];
//initiates the component
var ddlObject = new ej.dropdowns.DropDownList({
    //bind the data manager instance to dataSource property
    dataSource: sportsData ,
    //set the placeholder to DropDownList input
    placeholder:"Select games",
    //set the value to footer template
    footerTemplate:"<span class='foot'> Total list items: "+
sportsData.length + "</span>"
});

//render the component
ddlObject.appendTo('#ddlelement');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DropDownList</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:0 auto; width:250px;">

    <input type="text" id="ddlelement">
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

No records template

The DropDownList is provided with support to custom design the popup list content when no data is found and no matches found on search with the help of [noRecordsTemplate](#) property.

In the following sample, popup list content displays the notification of no data available.

INDEX.JS

```
//initiates the component
var ddlObject = new ej.dropdowns.DropDownList({
  //bind the data manager instance to dataSource property
  dataSource: [],
  //set the placeholder to DropDownList input
  placeholder: "Select an item",
  //set the value to noRecords template
  noRecordsTemplate: "<span class='norecord'> NO DATA AVAILABLE</span>"
});
//render the component
ddlObject.appendTo('#ddlelement');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DropDownList</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:0 auto; width:250px;">
    <input type="text" tabindex="1" id="ddlelement">
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Action failure template

There is also an option to custom design the popup list content when the data fetch request fails at the remote server. This can be achieved using the [actionFailureTemplate](#) property.

In the following sample, when the data fetch request fails, the DropDownList displays the notification.

INDEX.JS

```
//initiates the component
var ddlObject = new ej.dropdowns.DropDownList({
  //bind the data manager instance to dataSource property
  dataSource: new ej.data.DataManager({
    // Here, use the wrong url to display the action failure
    template
    url: 'https://services.odata.org/V4/Northwind/Northwind.svc/',
    adaptor: new ej.data.ODataV4Adaptor(),
    crossDomain: true
  }),
  //bind the Query instance to query property
  query: new
ej.data.Query().from('Employees').select(['FirstName']).take(6),
  //map the appropriate columns to fields property
  fields: { text: 'FirstName', value: 'EmployeeID' },
  //set the placeholder to DropDownList input
  placeholder: "Select an employee",
  //set the value to action failure template
  actionFailureTemplate: "<span class='action-failure'> Data fetch get
fails</span>"
});
ddlObject.appendTo('#ddlelement');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DropDownList</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>
```

```

<div id="container" style="margin:0 auto; width:250px;">
  <input type="text" tabindex="1" id="ddlelement">
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [How to achieve filtering](#)
- [How to group the data using header](#)
- [How to show the list items with icon](#)
- [How to render tooltip for the options](#)

Grouping in ##Platform_Name## Drop down list control

The DropDownList supports wrapping nested elements into a group based on different categories. The category of each list item can be mapped through the [groupBy](#) field in the data table. The group header is displayed both as inline and fixed headers. The fixed group header content is updated dynamically on scrolling the popup list with its category value.

In the following sample, vegetables are grouped according on its category using `groupBy` field.

INDEX.JS

```

var vegetableData = [
  { Vegetable: 'Cabbage', Category: 'Leafy and Salad', Id: 'item1' },
  { Vegetable: 'Spinach', Category: 'Leafy and Salad', Id: 'item2' },
  { Vegetable: 'Wheat grass', Category: 'Leafy and Salad', Id: 'item3' },
  { Vegetable: 'Yarrow', Category: 'Leafy and Salad', Id: 'item4' },
  { Vegetable: 'Pumpkins', Category: 'Leafy and Salad', Id: 'item5' },
  { Vegetable: 'Chickpea', Category: 'Beans', Id: 'item6' },
  { Vegetable: 'Green bean', Category: 'Beans', Id: 'item7' },
  { Vegetable: 'Horse gram', Category: 'Beans', Id: 'item8' },
  { Vegetable: 'Garlic', Category: 'Bulb and Stem', Id: 'item9' },
  { Vegetable: 'Nopal', Category: 'Bulb and Stem', Id: 'item10' },
  { Vegetable: 'Onion', Category: 'Bulb and Stem', Id: 'item11' }
];
//initiate the DropDownList
var vegetables = new ej.dropdowns.DropDownList({
  //set the grouped data to dataSource property
  dataSource: vegetableData,
  // map the groupBy field with Category column
  fields: { groupBy: 'Category', value: 'Vegetable' },
  // set the placeholder to the DropDownList input
  placeholder: "Find a vegetable"
});
//render the DropDownList component
vegetables.appendTo('#ddlelement');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DropDownList</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:0 auto; width:250px;">
    <br>
    <input type="text" id="ddlelement">
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

HTML select

The DropDownList also supports grouping of list items under specific groups by initiating the `<select>` element using `optgroup`. The nested items are wrapped based on the `<optgroup>` tag that is presents in the `<select>` element

```

<select id="selectElement">
  <optgroup label="Beans">
    <option value="1">Chickpea</option>
    <option value="2">Green bean</option>
    <option value="3">Horse gram</option>
  </optgroup>
  <optgroup label="Leafy and Salad">

```

```
<option value="4">Spinach</option>
<option value="5" selected="selected">Cabbage</option>
<option value="6">Wheat grass</option>
</optgroup>
</select>
,
```

INDEX.JS

```
// initialize DropDownList component
let ddlObject = new ej.dropdowns.DropDownList({
  placeholder: "Select a vegetable",
  popupHeight: "200px"
});
// render initialized DropDownList
ddlObject.appendTo('#selectElement');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DropDownList</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:0 auto; width:250px;">
    <select id="selectElement">
      <optgroup label="Beans">
        <option value="1">Chickpea</option>
        <option value="2">Green bean</option>
        <option value="3">Horse gram</option>
      </optgroup>
      <optgroup label="Leafy and Salad">
        <option value="4" selected="selected">Cabbage</option>
        <option value="5">Spinach</option>
        <option value="6">Wheat grass</option>
      </optgroup>
    </select>
  </div>
</body>
</html>
```

```

        </select>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customization

The grouping header is also provided with customization option. This allows custom designing using the [groupTemplate](#) property for both inline and fixed headers.

See Also

- [Group Template support to DropDownList.](#)
- [How to disable the fixed group header](#)

Filtering in ##Platform_Name## Drop down list control

The DropDownList has built-in support to filter data items when [allowFiltering](#) is enabled. The filter operation starts as soon as you start typing characters in the search box.

To display filtered items in the popup, filter the required data and return it to the DropDownList via [updateData](#) method by using the [filtering](#) event.

The following sample illustrates how to query the data source and pass the data to the DropDownList through the `updateData` method in `filtering` event.

INDEX.JS

```

//initiates the component
var ddlObject = new ej.dropdowns.DropDownList({
    //bind the data manager instance to dataSource property
    dataSource: new ej.data.DataManager({
        url:
        'https://services.odata.org/V4/Northwind/Northwind.svc/Customers',
        adaptor: new ej.data.ODATAV4Adaptor(),
        crossDomain: true
    }),
    //bind the Query instance to query property
    query: new ej.data.Query().select(['ContactName']),
    //map the appropriate columns to fields property
    fields: { value: 'ContactName' },
    //set the placeholder to DropDownList input
    placeholder: "Find a customer",
    //set the filterType to searching operation
    filterType: 'StartsWith',
    //sort the resulted items
    sortOrder: 'Ascending'
});
//render the component

```

```
ddlObject.appendTo('#ddlelement');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DropDownList</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:0 auto; width:250px;">
    <br>
    <input type="text" id="ddlelement">
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Limit the minimum filter character

When filtering the list items, you can set the limit for character count to raise remote request and fetch filtered data on the DropDownList. This can be done by manual validation within the filter event handler.

In the following example, the remote request does not fetch the search data until the search key contains three characters.

INDEX.JS

```
//initiates the component
var searchData = new ej.data.DataManager({
  url:
'https://services.odata.org/V4/Northwind/Northwind.svc/Customers',
  adaptor: new ej.data.ODataV4Adaptor(),
  crossDomain: true
```



```

});
var ddlObject = new ej.dropdowns.DropDownList({
  dataSource: searchData,
  query: new ej.data.Query().select(['ContactName',
  'CustomerID']).take(7),
  // map the appropriate column
  fields: { text: 'ContactName', value: 'CustomerID' },
  // set placeholder to DropDownList input element
  placeholder: "Select customers",
  //sort the resulted items
  sortOrder: 'Ascending',
  // set true to allowFiltering for enable filtering supports
  allowFiltering: true,
  //bind the filtering event handler
  filtering: (e) => {
    // load overall data when search key empty.
    if(e.text == '') e.updateData(searchData);
    else{
      // restrict the remote request until search key contains 3
characters.
      if (e.text.length < 3) { return; }
      var query = new ej.data.Query().select(['ContactName',
  'CustomerID']);
      query = (e.text !== '') ? query.where('ContactName', 'startswith',
e.text, true) : query;
      e.updateData(searchData, query);
    }
  }
});
//render the component
ddlObject.appendTo('#ddlelement');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DropDownList</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```

```

<div id="container" style="margin:0 auto; width:250px;">
  <br>
  <input type="text" id="ddlelement">
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Change the filter type

While filtering, you can change the filter type to `contains`, `startsWith`, or `endsWith` for string type within the filter event handler.

In the following examples, data filtering is done with `endsWith` type.

INDEX.JS

```

//initiates the component
var searchData = new ej.data.DataManager({
  url:
  'https://services.odata.org/V4/Northwind/Northwind.svc/Customers',
  adaptor: new ej.data.ODataV4Adaptor(),
  crossDomain: true
});
var ddlObject = new ej.dropdowns.DropDownList({
  dataSource: searchData,
  query: new ej.data.Query().select(['ContactName',
  'CustomerID']).take(7),
  // map the appropriate column
  fields: { text: 'ContactName', value: 'CustomerID' },
  // set placeholder to DropDownList input element
  placeholder: "Select customers",
  //sort the resulted items
  sortOrder: 'Ascending',
  // set true to allow filtering for enable filtering supports
  allowFiltering: true,
  //bind the filtering event handler
  filtering: (e) => {
    // load overall data when search key empty.
    if(e.text == '') e.updateData(searchData);
    else{
      var query = new ej.data.Query().select(['ContactName',
      'CustomerID']);
      query = (e.text !== '') ? query.where('ContactName', 'startswith',
      e.text, true) : query;
      e.updateData(searchData, query);
    }
  }
});
//render the component
ddlObject.appendTo('#ddlelement');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DropDownList</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:0 auto; width:250px;">
    <br>
    <input type="text" id="ddlelement">
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Case sensitive filtering

Data items can be filtered either with or without case sensitivity using the DataManager. This can be done by passing the fourth optional parameter of the **where** clause.

The following example shows how to perform case-sensitive filter.

INDEX.JS

```

//initiates the component
var searchData = new ej.data.DataManager({
  url:
    'https://services.odata.org/V4/Northwind/Northwind.svc/Customers',
  adaptor: new ej.data.ODataV4Adaptor(),
  crossDomain: true
});
var ddlObject = new ej.dropdowns.DropDownList({
  dataSource: searchData,
  query: new ej.data.Query().select(['ContactName',
    'CustomerID']).take(7),

```

```

// map the appropriate column
fields: { text: 'ContactName', value: 'CustomerID' },
// set placeholder to DropDownList input element
placeholder: "Select customers",
//sort the resulted items
sortOrder: 'Ascending',
// set true to allowFiltering for enable filtering supports
allowFiltering: true,
//bind the filtering event handler
filtering: (e) => {
    // load overall data when search key empty.
    if(e.text == '') e.updateData(searchData);
    else{
        var query = new ej.data.Query().select(['ContactName',
'CustomerID']);
        query = (e.text !== '') ? query.where('ContactName', 'startswith',
e.text, true) : query;
        e.updateData(searchData, query);
    }
};
//render the component
ddlObject.appendTo('#ddlelement');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DropDownList</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:0 auto; width:250px;">
    <br>
    <input type="text" id="ddlelement">
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";

```

```

}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Diacritics Filtering

The DropDownList supports diacritics filtering which will ignore the [diacritics](#) and makes it easier to filter the results in international characters lists when the [ignoreAccent](#) is enabled.

In the following sample, data with diacritics are bound as dataSource for DropDownList.

INDEX.JS

```

// create local data
var data = [
    'Aeróbics',
    'Aeróbics en Agua',
    'Aerografía',
    'Aeromodelaje',
    'Águilas',
    'Ajedrez',
    'Ala Delta',
    'Álbumes de Música',
    'Alusivos',
    'Análisis de Escritura a Mano'];
// initialize DropDownList component
var ddlObj = new ej.dropdowns.DropDownList({
    //set the local data to dataSource property
    dataSource: data,
    // set the placeholder to DropDownList input element
    placeholder: 'Select a value',
    // enabled the ignoreAccent property for ignore the diacritics
    ignoreAccent: true,
    // set true for enable the filtering support.
    allowFiltering: true,
    filterBarPlaceholder: 'e.g: aero'
});
ddlObj.appendTo('#ddlelement');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 DropDownList</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

```

```
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin:0 auto; width:250px;">
        <br>
        <input type="text" id="ddlelement">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

See Also

- [How to limit the search result while filtering](#)
- [How to highlight the matched characters in filtering](#)
- [How to modify the result data using remote data source](#)

Virtualization in DropDown List

Dropdown list virtualization is a technique used to efficiently render extensive lists of items while minimizing the impact on performance. This method is particularly advantageous when dealing with large datasets because it ensures that only a fixed number of DOM (Document Object Model) elements are created. When scrolling through the list, existing DOM elements are reused to display relevant data instead of generating new elements for each item. This recycling process is managed internally.

During virtual scrolling, the data retrieved from the data source depends on the popup height and the calculation of the list item height. Enabling the [enableVirtualization](#) option in a dropdown list activates this virtualization technique.

When fetching data from the data source, the [actionBegin](#) event is triggered before data retrieval begins. Then, the [actionComplete](#) event is triggered once the data is successfully fetched.

Furthermore, Incremental Search is supported with virtualization in the DropDownList component. When a key is typed, the focus is moved to the respective element, and the value is updated in the component in the open popup state. In the closed popup state, the respective value is updated in the component based on the typed key. The Incremental Search functionality is well-suited for scenarios involving remote data binding.

When the enableVirtualization property is enabled, the `skip` and `take` properties provided by the user within the Query class at the initial state or during the `actionBegin` or `actionComplete` events will not be considered, since it is internally managed and calculated based on certain dimensions with respect to the popup height.

Binding local data

The DropDownList can generate its list items through an array of complex data. For this, the appropriate columns should be mapped to the [fields](#) property. When using virtual scrolling, the list updates based on the scroll offset value, triggering a request to fetch more data from the server. As the data is being fetched, the `actionBegin` event occurs before the data retrieval starts. Once the data retrieval is successful, the `actionComplete` event is triggered, indicating that the data fetch process is complete.

In the following example, `id` column and `text` column from complex data have been mapped to the `value` field and `text` field, respectively.

INDEX.JS

```
var records = [];
for (var i = 1; i <= 150; i++) {
    var item = {
        id: 'id' + i,
        text: "Item " + i,
    };
    records.push(item);
}
//initiates the component
var ddlObject = new ej.dropdowns.DropDownList({
    //bind the dataSource property
    dataSource: records,
    //map the appropriate columns to fields property
    fields: { value: 'id', text: 'text' },
    //set the placeholder to DropDownList input
    placeholder: "Select an Item ",
    //set enableVirtualization property to true
    enableVirtualization: true,
    //set allowFiltering property to true
    allowFiltering: false,
    //set the height of the popup element
    popupHeight: '200px'
});
//render the component
ddlObject.appendTo('#ddlelement');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DropDownList</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-notifications/styles/material.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin:0 auto; width:250px;">
        <input type="text" id="ddlelement">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Binding Remote data

The DropDownList supports retrieval of data from remote data services with the help of **DataManager** component. When using remote data, it initially fetches all the data from the server, triggering the **actionBegin** and **actionComplete** events, and then stores the data locally. During virtual scrolling, additional data is retrieved from the locally stored data, triggering the **actionBegin** and **actionComplete** events at that time as well.

The following sample displays the OrderId from the **Orders** Data Service.

INDEX.JS

```

//initiates the component
var ddlObject = new ej.dropdowns.DropDownList({
    //bind the data source property
    dataSource: new ej.data.DataManager({
        url: 'https://ej2services.syncfusion.com/js/development/api/orders',
        adaptor: new ej.data.ODataV4Adaptor(),
        crossDomain: true
    }),
    //map the appropriate columns to fields property
    fields: { text: 'OrderID', value: 'OrderID' },
    //set the placeholder to DropDownList input
    placeholder: "Select an ID ",
    //set enableVirtualization property to true
    enableVirtualization: true,
    //set allowFiltering property to true
    allowFiltering: true,
    //set the height of the popup element
    popupHeight: '200px'
});
//render the component
ddlObject.appendTo('#ddlelement');

```


INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DropDownList</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:0 auto; width:250px;">
    <input type="text" id="ddlelement">
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Grouping with Virtualization

The DropDownList component supports grouping with Virtualization. It allows you to organize elements into groups based on different categories. Each item in the list can be classified using the [groupBy](#) field in the data table. After grouping, virtualization works similarly to local data binding, providing a seamless user experience. When the data source is bound to remote data, an initial request is made to retrieve all data for the purpose of grouping. Subsequently, the grouped data works in the same way as local data binding on virtualization.

The following sample shows the example for Grouping with Virtualization.

INDEX.JS

```

var records = [];
for (var i = 1; i <= 150; i++) {
  var dropdownsItem = {};
  dropdownsItem.id = 'id' + i;
  dropdownsItem.text = "Item ".concat(i);
  var randomAutoGroup = Math.floor(Math.random() * 4) + 1;

```

```

switch (randomAutoGroup) {
    case 1:
        dropdownsItem.group = 'Group D';
        break;
    case 2:
        dropdownsItem.group = 'Group C';
        break;
    case 3:
        dropdownsItem.group = 'Group B';
        break;
    case 4:
        dropdownsItem.group = 'Group A';
        break;
    default:
        break;
}
records.push(dropdownsItem);
}
//initiates the component
var ddlObject = new ej.dropdowns.DropDownList({
    //bind the data source property
    dataSource: records,
    //map the appropriate columns to fields property
    fields: { groupBy: 'group', text: 'text', value: 'id' },
    //set the placeholder to DropDownList input
    placeholder: "Select an Item ",
    //set enableVirtualization property to true
    enableVirtualization: true,
    //set allowFiltering property to true
    allowFiltering: true,
    //set the height of the popup element
    popupHeight: '200px'
});
//render the component
ddlObject.appendTo('#ddlelement');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 DropDownList</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin:0 auto; width:250px;">
        <input type="text" id="ddlelement">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Filtering with Virtualization

The DropDownList component supports Filtering with Virtualization. The DropDownList includes a built-in feature that enables data filtering when the [allowFiltering](#) option is enabled. In the context of Virtual Scrolling, the filtering process operates in response to the typed characters. Specifically, the DropDownList sends a request to the server, utilizing the full data source, to achieve filtering. Before initiating the request, an action event is triggered. Upon successful retrieval of data from the server, an action complete event is triggered. The initial data is loaded when the popup is opened. Whether the filter list has a selection or not, the popup closes.

The following sample shows the example for Filtering with Virtualization.

INDEX.JS

```

var records = [];
for (var i = 1; i <= 150; i++) {
    var item = {
        id: 'id' + i,
        text: "Item " + i,
    };
    records.push(item);
}
//initiates the component
var ddlObject = new ej.dropdowns.DropDownList({
    //bind the data source property
    dataSource: records,
    //map the appropriate columns to fields property
    fields: { value: 'id', text: 'text' },
    //set the placeholder to DropDownList input
    placeholder: "Select an Item ",
    //set enableVirtualization property to true
    enableVirtualization: true,
    //set allowFiltering property to true
    allowFiltering: true,
    //set the height of the popup element
    popupHeight: '200px'
});

```

```
//render the component
ddlObject.appendTo('#ddlelement');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DropDownList</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:0 auto; width:250px;">
    <input type="text" id="ddlelement">
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Localization in ##Platform_Name## Drop down list control

The Localization library allows you to localize static text content of the [noRecordsTemplate](#)

and [actionFailureTemplate](#) properties according to the culture currently assigned to the DropDownList.

| Locale key | en-US (default) |

|-----|-----|

| noRecordsTemplate | No records found |

| actionFailureTemplate | The request failed |

Loading translations

To load translation object to your application, use load function of the **L10n** class.

In the following sample, French culture is set to the DropDownList and no data is loaded. Hence, the [noRecordsTemplate](#) property displays its text in French culture initially, and if the sample is run offline, the [actionFailureTemplate](#) property displays its text appropriately.

INDEX.JS

```
//bind the data manager instance to dataSource property
var customerData = new ej.data.DataManager({
  url: 'https://services.odata.org/V4/Northwind/Northwind.svc/Customers',
  adaptor: new ej.data.ODataV4Adaptor(),
  crossDomain: true
});
var ddlObj = new ej.dropdowns.DropDownList({
  dataSource: customerData,
  // set locale culture to DropDownList
  locale: 'fr-BE',
  // map appropriate column
  fields: { text: 'ContactName', value: 'CustomerID' },
  // take 0 item to showcase noRecordsTemplate property.
  query: new ej.data.Query().select(['ContactName',
    'CustomerID']).take(0),
  // set placeholder to DropDownList input element
  placeholder: 'Sélectionnez un éléments'
});
ddlObj.appendTo('#ddlelement');
ej.base.L10n.load({
  'fr-BE': {
    'dropdowns': {
      'noRecordsTemplate': "Aucun enregistrement trouvé",
      'actionFailureTemplate': "Modèle d'échec d'action"
    }
  }
});
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DropDownList</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin:0 auto; width:250px;">
        <br>
        <input type="text" id="ddlelement">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Accessibility](#)
- [How to bind the data to the combobox](#)

Style in ##Platform_Name## Drop down list control

The following content provides the exact CSS structure that can be used to modify the control's appearance based on the user preference.

Customizing the appearance of wrapper element

Use the following CSS to customize the appearance of wrapper element.

```

.e-ddl.e-input-group.e-control-wrapper .e-input {
font-size: 20px;
font-family: emoji;
color: #ab3243;
background: #32a5ab;
}

```

Customizing the dropdown icon's color

Use the following CSS to customize the dropdown icon's color.

```

.e-ddl.e-input-group .e-input-group-icon,.e-ddl.e-input-group.e-control-wrapper .e-input-group-
icon:hover {
color: #bb233d;
font-size: 13px;
}

```

```
}
,
```

Customizing the focus color

Use the following CSS to customize the focusing color of input element.

```
,
```

```
.e-ddl.e-input-group.e-control-wrapper.e-input-focus::before, .e-ddl.e-input-group.e-control-wrapper.e-input-focus::after {
```

```
background: #c000ff;
```

```
}
```

```
,
```

Customizing the outline theme's focus color

Use the following CSS to customize the focusing color of outline theme.

```
,
```

```
.e-outline.e-input-group.e-input-focus:hover:not(.e-success):not(.e-warning):not(.e-error):not(.e-disabled):not(.e-float-icon-left),.e-outline.e-input-group.e-input-focus.e-control-wrapper:hover:not(.e-success):not(.e-warning):not(.e-error):not(.e-disabled):not(.e-float-icon-left),.e-outline.e-input-group.e-input-focus:not(.e-success):not(.e-warning):not(.e-error):not(.e-disabled),.e-outline.e-input-group.e-control-wrapper.e-input-focus:not(.e-success):not(.e-warning):not(.e-error):not(.e-disabled) {
```

```
border-color: #b1bd15;
```

```
box-shadow: inset 1px 1px #b1bd15, inset -1px 0 #b1bd15, inset 0 -1px #b1bd15;
```

```
}
```

```
,
```

Customizing the disabled component's text color

Use the following CSS to customize the text color when the component is disabled.

```
,
```

```
.e-input-group.e-control-wrapper .e-input[disabled] {
```

```
-webkit-text-fill-color: #0d9133;
```

```
}
```

```
,
```

Customizing the float label element's focusing color

Use the following CSS to customize the focusing color of float label element.

```
,
```

```
.e-float-input.e-input-group:not(.e-float-icon-left) .e-float-line::before,.e-float-input.e-control-wrapper.e-input-group:not(.e-float-icon-left) .e-float-line::before,.e-float-input.e-input-group:not(.e-float-icon-left) .e-float-line::after,.e-float-input.e-control-wrapper.e-input-group:not(.e-float-icon-left) .e-float-line::after {
```

```
background-color: #2319b8;
```

```

}
.e-ddl.e-lib.e-input-group.e-control-wrapper.e-float-input.e-input-focus .e-float-text.e-label-top {
color: #2319b8;
}
、

```

Customizing the color of the placeholder text

Use the following CSS to customize the text color of placeholder.

```

、
.e-ddl.e-input-group input.e-input::placeholder {
color: red;
}
、

```

Customizing the background color of focus, hover, and active item's

Use the following CSS to customize the background color of focus, hover and active item's.

```

、
.e-dropdownbase .e-list-item.e-item-focus, .e-dropdownbase .e-list-item.e-active, .e-dropdownbase .e-
list-item.e-active.e-hover, .e-dropdownbase .e-list-item.e-hover {
background-color: #1f9c99;
color: #2319b8;
}
、

```

Customizing the appearance of pop-up element

Use the following CSS to customize the appearance of popup element.

```

、
.e-dropdownbase .e-list-item, .e-dropdownbase .e-list-item.e-item-focus {
background-color: #29c2b8;
color: #207cd9;
font-family: emoji;
min-height: 29px;
}
、

```

Adding mandatory asterisk to placeholder and float label

You can add a mandatory asterisk(*) to placeholder and float label using `.e-input-group.e-control-wrapper.e-float-input .e-float-text::after` class.

INDEX.JS


```

var sportsData = ['Badminton', 'Cricket', 'Football', 'Golf', 'Tennis'];
// initialize DropDownList component
var listObj = new ej.dropdowns.DropDownList({
  dataSource: sportsData,
  // set placeholder to DropDownList input element
  placeholder: "Select games"});
listObj.appendTo('#ddlelement');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DropDownList</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="asterisk.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:0 auto; width:250px;">
    <input type="text" tabindex="1" id="ddlelement">
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Accessibility in `##Platform_Name##` Drop down list control

The DropDownList component has been designed, keeping in mind the WAI-ARIA specifications, and applies the WAI-ARIA roles, states, and properties along with keyboard support. This component is characterized by complete keyboard interaction support and ARIA accessibility support that makes it easy for people who use assistive technologies (AT) or those who completely rely on keyboard navigation.

The DropDownList component followed the accessibility guidelines and standards, including [ADA](#), [Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the DropDownList component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2 Support](#) | |

| [Section 508 Support](#) | |

| [Screen Reader Support](#) | |

| [Right-To-Left Support](#) | |

| [Color Contrast](#) | |

| [Mobile Device Support](#) | |

| [Keyboard Navigation Support](#) | |

| [Accessibility Checker Validation](#) | |

| [Axe-core Accessibility Validation](#) | |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<div> - All features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>

WAI-ARIA attributes

The DropDownList component uses the **Listbox** role, and each list item has an **option** role. The following **ARIA attributes** denote the DropDownList state.

| **Properties** | **Functionalities** |

| --- | --- |

| aria-haspopup | Indicates whether the DropDownList input element has a popup list or not. |

| aria-expanded | Indicates whether the popup list has expanded or not. |

| aria-selected | Indicates the selected option. |

| aria-readonly | Indicates the readonly state of the DropDownList element. |

| aria-disabled | Indicates whether the DropDownList component is in a disabled state or not. |

| aria-activedescendent | This attribute holds the ID of the active list item to focus its descendant child element. |

| aria-owns | This attribute contains the ID of the popup list to indicate popup as a child element. |

Keyboard interaction

You can use the following key shortcuts to access the DropDownList without interruptions.

| **Keyboard shortcuts** | **Actions** |

| --- | --- |

| Arrow Down | Selects the first item in the DropDownList when no item selected. Otherwise, selects the item next to the currently selected item. |

| Arrow Up | Selects the item previous to the currently selected one. |

| Page Down | Scrolls down to the next page and selects the first item when popup list opens. |

| Page Up | Scrolls up to the previous page and selects the first item when popup list opens. |

| Enter | Selects the focused item, and when it is in an open state the popup list closes. Otherwise, toggles the popup list. |

| Tab | Focuses on the next TabIndex element on the page when the popup is closed. Otherwise, closes the popup list and remains the focus of the component. |

| Shift + tab | Focuses on the previous TabIndex element on the page when the popup is closed. Otherwise, closes the popup list and remains the focus of the component. |

| Alt + Down | Opens the popup list. |

| Alt + Up | Closes the popup list. |

| Esc(Escape) | Closes the popup list when it is in an open state and the currently selected item remains the same. |

| Home | Selects the first item. |

| End | Selects the last item. |

In the below sample, focus the DropDownList component using alt+t keys.

INDEX.JS

```
var gameList = [
    { id: 'Game1', game: 'Badminton' },
    { id: 'Game2', game: 'Basketball' },
```

```

        { id: 'Game3', game: 'Cricket' },
        { id: 'Game4', game: 'Football' },
        { id: 'Game5', game: 'Golf' },
        { id: 'Game6', game: 'Hockey' },
        { id: 'Game7', game: 'Rugby' },
        { id: 'Game8', game: 'Snooker' },
        { id: 'Game9', game: 'Tennis' },
    ];
    // initialize DropDownList component
    var ddlObject = new ej.dropdowns.DropDownList({
        //set the data to dataSource property
        dataSource: gameList,
        //map to column to fields
        fields: { text: 'game', value: 'id' },
        // set placeholder to DropDownList input element
        placeholder: "Select games",
        // set the popup list height
        popupHeight: '200px'
    });
    // render initialized DropDownList
    ddlObject.appendTo('#ddlelement');
    document.onkeyup = function (e) {
        if (e.altKey && e.keyCode === 84 /* t */) {
            // press alt+t to focus the control.
            ddlObject.focusIn();
        }
    };
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 DropDownList</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin:0 auto; width:250px;">
        <input type="text" tabindex="1" id="ddlelement">
    </div>
</body>
</html>

```

```

var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Ensuring accessibility

The DropDownList component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the DropDownList component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the DropDownList component with accessibility tools.

See also

- [Accessibility in Syncfusion components](#)

How To

Add item in ##Platform_Name## Drop down list control

You can add item in between based on item [index](#). If you add new item without item index, item will be added as last item in list.

The following example demonstrate how to add item in between in DropDownList.

INDEX.JS

```

var sportsData = [
  { Id: 'game1', Game: 'Badminton' },
  { Id: 'game2', Game: 'Football' },
  { Id: 'game3', Game: 'Tennis' }
];
//initiate the DropDownList
var dropDownListObject = new ej.dropdowns.DropDownList({
  // bind the sports Data to datasource property
  dataSource: sportsData,
  // maps the appropriate column to fields property
  fields: { text: 'Game', value: 'Id' },
  //set the placeholder to DropDownList input
  placeholder: "Select a game"
});
//render the component
dropDownListObject.appendTo('#ddlelement');

// add item at first
document.getElementById('first').onclick = () => {
  dropDownListObject.addItem({ Id: 'game0', Game: 'Hockey' }, 0);
};
// add item in between
document.getElementById('between').onclick = () => {
  dropDownListObject.addItem({ Id: 'game4', Game: 'Golf' }, 2);
};

```

```
// add item at last
document.getElementById('last').onclick = () => {
    dropDownListObject.addItem({ Id: 'game5', Game: 'Cricket' });
};
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DropDownList</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:0 auto; width:250px;">
    <br>
    <input type="text" id="ddlelement">

    <div style="padding: 50px 0">
      <button id="first" class="e-control e-btn"> add item (Hockey) in
first</button>
    </div>
    <div style="padding-left: 50px 0">
      <button id="between" class="e-control e-btn"> add item (Golf) in
between</button>
    </div>
    <div style="padding: 50px 0">
      <button id="last" class="e-control e-btn"> add item (Cricket) in
last</button>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Cascading in ##Platform_Name## Drop down list control

The cascading DropDownList is a series of DropDownList, where the value of one DropDownList depends upon another's value. This can be configured by using the [change](#) event of the parent DropDownList. Within that change event handler, data has to be loaded to the child DropDownList based on the selected value of the parent DropDownList.

The following example, shows the cascade behavior of country, state, and city DropDownList. Here, the [dataBind](#) method is used to reflect the property changes immediately to the DropDownList.

INDEX.JS

```
//define the country DropDownList data
var countryData = [
    { CountryName: 'Australia', CountryId: '2' },
    { CountryName: 'United States', CountryId: '1' }
];
//define the state DropDownList data
var stateData = [
    { StateName: 'New York', CountryId: '1', StateId: '101' },
    { StateName: 'Virginia ', CountryId: '1', StateId: '102' },
    { StateName: 'Tasmania ', CountryId: '2', StateId: '105' }
];
//define the city DropDownList data
var cityData = [
    { CityName: 'Albany', StateId: '101', CityId: 201 },
    { CityName: 'Beacon ', StateId: '101', CityId: 202 },
    { CityName: 'Emporia', StateId: '102', CityId: 206 },
    { CityName: 'Hampton ', StateId: '102', CityId: 205 },
    { CityName: 'Hobart', StateId: '105', CityId: 213 },
    { CityName: 'Launceston ', StateId: '105', CityId: 214 }
];
//initiates the country DropDownList
var countryObj = new ej.dropdowns.DropDownList({
    //set the data to dataSource property
    dataSource: countryData,
    fields: { value: 'CountryId', text: 'CountryName' },
    //bind the change event handler
    change: () => {
        //Query the data source based on country DropDownList selected value
        stateObj.query = new ej.data.Query().where('CountryId', 'equal',
countryObj.value);
        // enable the state DropDownList
        stateObj.enabled = true;
        //clear the existing selection.
        stateObj.text = null;
        // bind the property changes to state DropDownList
        stateObj.dataBind();
        //clear the existing selection in city DropDownList
        cityObj.text = null;
        //disabe the city DropDownList
        cityObj.enabled = false;
        //bind the property cahnges to City DropDownList
        cityObj.dataBind();
    },
    placeholder: 'Select a country',
});
```

```

//render the country DropDownList
countryObj.appendTo('#countries');
//initiates the state DropDownList
var stateObj = new ej.dropdowns.DropDownList({
    dataSource: stateData,
    fields: { value: 'StateId', text: 'StateName' },
    // set disable state by default to prevent user interact.
    enabled: false,
    change: () => {
        // Query the data source based on state DropDownList selected value
        cityObj.query = new ej.data.Query().where('StateId', 'equal',
stateObj.value);
        // enable the city DropDownList
        cityObj.enabled = true;
        //clear the existing selection
        cityObj.text = null;
        // bind the property change to city DropDownList
        cityObj.dataBind();
    },
    placeholder: 'Select a state',
});
//render the state DropDownList
stateObj.appendTo('#states');
//initiates the city DropDownList
var cityObj = new ej.dropdowns.DropDownList({
    dataSource: cityData,
    fields: { text: 'CityName', value: 'CityId' },
    // disable the DropDownList by default to prevent the user interact.
    enabled: false,
    placeholder: 'Select a city',
});
//render the city DropDownList
cityObj.appendTo('#cities');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DropDownList</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>

```



```

<body>

  <div id="container" style="margin:50px auto 0; width:250px;">
    <br>
    <input type="text" id="countries">
    <div class="padding-top">
      <input type="text" id="states">
    </div>
    <div class="padding-top">
      <input type="text" id="cities">
    </div>
  </div>
</body>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Clear item in ##Platform_Name## Drop down list control

You can clear the selected item in the below two different ways.

By clicking on the **clear icon** which is shown in DropDownList element, you can clear the selected item in DropDownList through **interaction**. By using [showClearButton](#) property, you can enable the clear icon in DropDownList element.

Through **programmatic** you can set **null** value to anyone of the index, text or value property to clear the selected item in DropDownList.

The following example demonstrate about how to clear the selected item in DropDownList.

INDEX.JS

```

var sportsData = ['Badminton', 'Cricket', 'Football', 'Golf', 'Tennis'];
// initialize DropDownList component
var dropDownListObject = new ej.dropdowns.DropDownList({
  //set the data to dataSource property
  dataSource: sportsData,
  // set placeholder to DropDownList input element
  placeholder: "Select a game",
  //enable the clear icon
  showClearButton: true
});
// render initialized DropDownList
dropDownListObject.appendTo('#ddelement');

// Set null value to value property for clear the selected item
document.getElementById('btn').onclick = () => {
  dropDownListObject.value = null;
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>

```

```

<title>Essential JS 2 DropDownList</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="styles.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin:0 auto; width:250px;">
        <br>
        <input type="text" id="ddlelement">
    </div>
    <div style="padding: 50px">
        <button id="btn" class="e-control e-btn"> Set null to value
property</button>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Close popup in ##Platform_Name## Drop down list control

By using the `hidePopup` method in DropDownList, you can close the popup on scroll when triggered the windows scroll event.

The following example demonstrate about how to close the popup on scroll.

INDEX.JS

```

var sportsData = ['Badminton', 'Cricket', 'Football', 'Golf', 'Tennis'];
// initialize DropDownList component
var dropDownListObject = new ej.dropdowns.DropDownList({
    //set the data to dataSource property
    dataSource: sportsData,
    // set placeholder to DropDownList input element
    placeholder: "Select a game"
});
// render initialized DropDownList
dropDownListObject.appendTo('#ddlelement');

```

```
// bind the onscroll event to window
window.onscroll = () => {
    dropDownListObject.hidePopup();
};
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DropDownList</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div style="padding: 50px">
    <h4> You can close the opened popup by scroll the page.</h4>
  </div>
  <div id="container" style="margin:0 auto; width:250px;">
    <br>
    <input type="text" id="ddlelement">
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Group header in ##Platform_Name## Drop down list control

The following example demonstrate about how to disable the Fixed group header in DropDownList through CSS by using `visibility` attribute.

INDEX.JS

```
var vegetableData = [
  { Vegetable: 'Cabbage', Category: 'Leafy and Salad', Id: 'item1' },
  { Vegetable: 'Spinach', Category: 'Leafy and Salad', Id: 'item2' },
```

```

    { Vegetable: 'Wheat grass', Category: 'Leafy and Salad', Id: 'item3' },
    { Vegetable: 'Yarrow', Category: 'Leafy and Salad', Id: 'item4' },
    { Vegetable: 'Pumpkins', Category: 'Leafy and Salad', Id: 'item5' },
    { Vegetable: 'Chickpea', Category: 'Beans', Id: 'item6' },
    { Vegetable: 'Green bean', Category: 'Beans', Id: 'item7' },
    { Vegetable: 'Horse gram', Category: 'Beans', Id: 'item8' },
    { Vegetable: 'Garlic', Category: 'Bulb and Stem', Id: 'item9' },
    { Vegetable: 'Nopal', Category: 'Bulb and Stem', Id: 'item10' },
    { Vegetable: 'Onion', Category: 'Bulb and Stem', Id: 'item11' }
  ];
  //initiate the DropDownList
  var vegetables = new ej.dropdowns.DropDownList({
    //set the grouped data to dataSource property
    dataSource: vegetableData,
    // map the groupBy field with Category column
    fields: { groupBy: 'Category', text: 'Vegetable', value: 'Id' },
    // set the placeholder to the DropDownList input
    placeholder: "Select a vegetable",
    // Set the popup list height
    popupHeight: '200px'
  });
  //render the DropDownList component
  vegetables.appendTo('#ddlelement');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DropDownList</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:0 auto; width:250px;">
    <br>
    <input type="text" id="ddlelement">
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";

```

```

}
    </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Highlight filtering in ##Platform_Name## Drop down list control

By using the **highlightSearch** method, you can highlight the matched character in DropDownList filtering.

The following example demonstrates about how to highlight the matched character in filtering.

INDEX.JS

```

//bind the data manager instance to dataSource property
var data=new ej.data.DataManager({
    url:
    'https://services.odata.org/V4/Northwind/Northwind.svc/Customers',
    adaptor: new ej.data.ODataV4Adaptor(),
    crossDomain: true
});
//initiates the component
var customers = new ej.dropdowns.DropDownList({
    //bind the data manager instance to dataSource property
    dataSource : data,
    //bind the Query instance to query property
    query: new ej.data.Query().select(['ContactName']).take(6),
    //map the appropriate columns to fields property
    fields: { value: 'ContactName'},
    //set the placeholder to DropDownList input
    placeholder: "Select a name",
    // set true to allowFiltering for enable filtering supports
    allowFiltering: true,
    //bind the filtering event handler
    filtering: (e) => {
        // take text for highlight the character in list items.
        let text = e.text;
        let query = new ej.data.Query().select(['ContactName',
        'CustomerID']);
        //frame the query based on search string with filter type.
        query = (e.text !== '') ? query.where('ContactName', 'startswith',
        e.text, true) : query;
        //pass the filter data source, filter query to updateData method.
        e.updateData(data, query);
        setTimeout(() => {
            let popupElement = document.getElementById('ddlelement_popup');
            // get the list from popup element
            let lists = popupElement.querySelector('ul');
            // pass the element, text, ignore case and filter type as
            argument to highlightSearch method.
            ej.dropdowns.highlightSearch(lists, text, true, 'StartsWith');
        }, 300);
    }
});
//render the component
customers.appendTo('#ddlelement');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DropDownList</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:0 auto; width:250px;">
    <br>
    <input type="text" id="ddlelement">
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Icons support in ##Platform_Name## Drop down list control

You can render **icons** to the list items by mapping the [iconCss](#) field. This `iconCss` field create a span in the list item with mapped class name to allow styling as per your need.

In the following sample, icon classes are mapped with `iconCss` field.

INDEX.JS

```

var sortFormatData = [
  { Class: 'asc-sort', Type: 'Sort A to Z', Id: '1' },
  { Class: 'dsc-sort', Type: 'Sort Z to A ', Id: '2' },
  { Class: 'filter', Type: 'Filter', Id: '3' },
  { Class: 'clear', Type: 'Clear', Id: '4' }
];
//initiates the component
var sortFormat = new ej.dropdowns.DropDownList({
  //set the data to dataSource property
  dataSource: sortFormatData,
  // map the icon column to iconCSS field.

```

```

    fields: { value: 'Type', iconCss: 'Class' },
    // set placeholder to DropDownList input element
    placeholder: 'Find a format'
  });
  sortFormat.appendTo('#ddlelement');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DropDownList</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:0 auto; width:250px;">
    <br>
    <input type="text" id="ddlelement">
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Incremental search in ##Platform_Name## Drop down list control

DropDownList supports incremental search, by default. You can search the list item by focusing the DropDownList and typing the characters in it. The closely matched items are selected sequentially.

If the same key is searched once again, the next matched item is selected.

Modify data in ##Platform_Name## Drop down list control

When binding the remote data source, by using the [actionComplete](#) event, you can modify the result data before passing it to DropDownList.

The following sample demonstrate how to modify the result data.

INDEX.JS

```

var ddlObject = new ej.dropdowns.DropDownList({
    //bind the data manager instance to dataSource property
    dataSource: new ej.data.DataManager({
        url:
        'https://services.odata.org/V4/Northwind/Northwind.svc/Customers',
        adaptor: new ej.data.ODataV4Adaptor(),
        crossDomain: true
    }),
    //bind the Query instance to query property
    query: new ej.data.Query().select(['ContactName']),
    //map the appropriate columns to fields property
    fields: { value: 'ContactName'},
    //set the placeholder to DropDownList input
    placeholder: "Find a customer",
    //sort the resulted items
    actionComplete: function (e) {
        // initially result contains 6 items
        console.log("Before modified the result: " + e.result.length);
        // remove first 2 items from result.
        e.result.splice(0, 2);
        // now displays the result count is 4.
        console.log("After modified the result: " + e.result.length);
    }
});
//render the component
ddlObject.appendTo('#ddlelement');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 DropDownList</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin:0 auto; width:250px;">
        <br>
        <input type="text" id="ddlelement">

```



```

    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Multiple cascading in ##Platform_Name## Drop down list control

The following example demonstrate about how to preselect the list items in multiple cascading DropDownList.

INDEX.JS

```

var countryData = [
    { CountryName: 'United States', CountryId: '1' },
    { CountryName: 'Australia', CountryId: '2' }
];
//define the state DropDownList data
var stateData = [
    { StateName: 'New York', CountryId: '1', StateId: '101' },
    { StateName: 'Virginia ', CountryId: '1', StateId: '102' },
    { StateName: 'Tasmania ', CountryId: '2', StateId: '105' }
];
//define the city DropDownList data
var cityData = [
    { CityName: 'Albany', StateId: '101', CityId: 201 },
    { CityName: 'Beacon ', StateId: '101', CityId: 202 },
    { CityName: 'Hampton ', StateId: '102', CityId: 205 },
    { CityName: 'Emporia', StateId: '102', CityId: 206 },
    { CityName: 'Hobart', StateId: '105', CityId: 213 },
    { CityName: 'Launceston ', StateId: '105', CityId: 214 }
];
//initiates the country DropDownList
var countryObj = new ej.dropdowns.DropDownList({
    dataSource: countryData,
    // map the appropriate the column to fields property
    fields: { value: 'CountryId', text: 'CountryName' },
    //bind the change event handler
    change: countryChange,
    index: 1,
    // set the placeholder to the DropDownList input
    placeholder: 'Select a country',
});
//render the country DropDownList
countryObj.appendTo('#countries');
//initiates the state DropDownList
var stateObj = new ej.dropdowns.DropDownList({
    dataSource: stateData,
    // map the appropriate the column to fields property
    fields: { value: 'StateId', text: 'StateName' },
    // set disable state by default to prevent user interact.
    enabled: false,

```

```

        //bind the change event handler
        change: stateChange,
        // set the placeholder to the DropDownList input
        placeholder: 'Select a state',
    });
    //render the state DropDownList
    stateObj.appendTo('#states');
    //initiates the city DropDownList
    var cityObj = new ej.dropdowns.DropDownList({
        dataSource: cityData,
        // map the appropriate the column to fields property
        fields: { text: 'CityName', value: 'CityId' },
        // disable the DropDownList by default to prevent the user interact.
        enabled: false,
        // set the placeholder to the DropDownList input
        placeholder: 'Select a city',
    });
    //render the city DropDownList
    cityObj.appendTo('#cities');
    // initially change event not triggered, so manually call the corresponding
    function
    countryChange();
    // preselect an item from filtered state DropDownList
    stateObj.index = 0;
    stateObj.dataBind();
    // initially change event not triggered, so manually call the corresponding
    function
    stateChange();
    // preselect an item from filtered city DropDownList
    cityObj.index = 0;
    cityObj.dataBind();
    function stateChange() {
        // Query the data source based on state DropDownList selected value
        cityObj.query = new ej.data.Query().where('StateId', 'equal',
stateObj.value);
        // enable the city DropDownList
        cityObj.enabled = true;
        //clear the existing selection
        cityObj.text = null;
        // bind the property change to city DropDownList
        cityObj.dataBind();
    }
    function countryChange() {
        //Query the data source based on country DropDownList selected value
        stateObj.query = new ej.data.Query().where('CountryId', 'equal',
countryObj.value);
        // enable the state DropDownList
        stateObj.enabled = true;
        //clear the existing selection.
        stateObj.text = null;
        // bind the property changes to state DropDownList
        stateObj.dataBind();
        //clear the existing selection in city DropDownList
        cityObj.text = null;
        //disabe the city DropDownList
        cityObj.enabled = false;
        //bind the property cahnges to City DropDownList

```

```
cityObj.dataBind();
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DropDownList</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:50px auto 0; width:250px;">
    <br>
    <input type="text" id="countries">
    <div class="padding-top">
      <input type="text" id="states">
    </div>
    <div class="padding-top">
      <input type="text" id="cities">
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Remote data bind in ##Platform_Name## Drop down list control

Before component rendering, you can get the total items count by using [actionComplete](#) event with its result arguments. After rendering this component, you can get the total items count by using [getItems](#) method.

The following example demonstrate how to get the total items count.

INDEX.JS

```

var customers = new ej.dropdowns.DropDownList({
    //bind the data manager instance to dataSource property
    dataSource: new ej.data.DataManager({
        url:
        'https://services.odata.org/V4/Northwind/Northwind.svc/Customers',
        adaptor: new ej.data.ODataV4Adaptor(),
        crossDomain: true
    }),
    //bind the Query instance to query property
    query: new ej.data.Query().select(['ContactName']).take(6),
    //map the appropriate columns to fields property
    fields: { value: 'ContactName'},
    //set the placeholder to DropDownList input
    placeholder: "Find a customer",
    //sort the resulted items
    aactionComplete: function (e) {
        // get total items count
        console.log("Total items count: " + e.result.length);
        let element = document.createElement('p');
        element.innerText = "Total items count: " + e.result.length;
        document.getElementById('event').append(element);
    }
});
//render the component
customers.appendTo('#ddlelement');
document.getElementById('btn').onclick = () => {
    // get items count using getItems method
    console.log("Total items count: " + customers.getItems().length);
    let element = document.createElement('p');
    element.innerText = "Total items count: " + customers.getItems().length;
    document.getElementById('event').append(element);
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DropDownList</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

```

```

    <div id="container" style="margin:0 auto; width:250px;">
        <br>
        <input type="text" id="ddlelement">
        <div style="margin: 50px">
            <button id="btn" class="e-btn e-control"> Get items</button>
        </div>
        <p id="event"> </p>
    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Remove item in ##Platform_Name## Drop down list control

The following example demonstrate about how to remove an item from DropDownList.

INDEX.JS

```

var sportsData = [
    { Id: 'game1', Game: 'Badminton' },
    { Id: 'game2', Game: 'Football' },
    { Id: 'game3', Game: 'Tennis' }
];
//initiate the DropDownList
var dropDownListObject = new ej.dropdowns.DropDownList({
    // bind the sports Data to datasource property
    dataSource: sportsData,
    // maps the appropriate column to fields property
    fields: { text: 'Game', value: 'Id' },
    //set the placeholder to DropDownList input
    placeholder: "Select a game"
});
//render the componen
dropDownListObject.appendTo('#ddlelement');

document.getElementById('first').onclick = () => {
    // create DropDownList object
    var obj = document.getElementById('ddlelement');
    if (obj.ej2_instances[0].list) {
        // Remove the selected value if 0th index selected
        if (dropDownListObject.index === 0) {
            dropDownListObject.value = null;
            dropDownListObject.dataBind();
        }
        // remove first item in list
        (obj.ej2_instances[0].list.querySelectorAll('li')[0]).remove();
        if (!obj.ej2_instances[0].list.querySelectorAll('li')[0]) {
            dropDownListObject.dataSource = [];
            // enable the nodata template when no data source is empty.
            obj.ej2_instances[0].list.classList.add('e-nodata');
        }
    }
}

```

```

    }
    else {
        // remove first item in list
        dropDownListObject.dataSource.splice(0, 1);
    }
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DropDownList</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:0 auto; width:250px;">
    <br>
    <input type="text" id="ddlelement">
    <div style="padding: 50px 0">
      <button id="first" class="e-control e-btn"> Remove 0th
item</button>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Search on filtering in ##Platform_Name## Drop down list control

The following example demonstrates about how to set limit the search result on filtering.

INDEX.JS

```

var data=new ej.data.DataManager({
  url:
  'https://services.odata.org/V4/Northwind/Northwind.svc/Customers',

```

```

        adaptor: new ej.data.ODataV4Adaptor(),
        crossDomain: true
    });
var customers = new ej.dropdowns.DropDownList({
    //bind the data manager instance to dataSource property
    dataSource : data,
    //bind the Query instance to query property
    query: new ej.data.Query().select(['ContactName']).take(6),
    //map the appropriate columns to fields property
    fields: { value: 'ContactName'},
    //set the placeholder to DropDownList input
    placeholder: "Select a name",
    // sort the resulted items
    sortOrder: 'Ascending',
    // set true to allowFiltering for enable filtering supports
    allowFiltering: true,
    // bind the filtering event handler
    filtering: (e) => {
        // set limit as 4 to search result
        let query = new ej.data.Query().select(['ContactName']).take(6);
        query = (e.text !== '') ? query.where('ContactName', 'startswith',
e.text, true) : query;
        e.updateData(data, query);
    }
});
//render the component
customers.appendTo('#ddlelement');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 DropDownList</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin:0 auto; width:250px;">
        <br>
        <input type="text" id="ddlelement">
    </div>

```

```

<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Tooltip in ##Platform_Name## Drop down list control

You can achieve this behavior by using **ej2-tooltip** component. When the mouse hover on the DropDownList option that tooltip display some details related to hovered list item.

INDEX.JS

```

//Initialize DropDownList component
var listObj = new ej.dropdowns.DropDownList({
    // define the array of JSON data
    dataSource: [
        { id: '1', text: 'Australia', content: 'National sports is Cricket' },
        { id: '2', text: 'Bhutan', content: 'National sports is Archery' },
        { id: '3', text: 'China', content: 'National sports is Table Tennis' },
        { id: '4', text: 'Cuba', content: 'National sports is Baseball' },
        { id: '5', text: 'India', content: 'National sports is Hockey' },
        { id: '6', text: 'Spain', content: 'National sports is Football' },
        { id: '7', text: 'United States', content: 'National sports is Baseball' }
    ],
    // map the appropriate column to fields property
    fields: { text: 'text', tooltip: 'id' },
    // set the placeholder to the DropDownList input
    placeholder: 'Select a country',
    // bind the DropDownList close event
    close: () => { tooltip.close(); }
});
listObj.appendTo('#ddltooltip');
//Initialize Tooltip component
var tooltip = new ej.popups.Tooltip({
    // default content of tooltip
    content: 'Loading...',
    // set target element to tooltip
    target: '.e-list-item',
    // set position of tooltip
    position: 'top center',
    // bind beforeRender event
    beforeRender: onBeforeRender
});
tooltip.appendTo('body');
function onBeforeRender(args) {
    // get the target element
    var listElement = document.getElementById('ddltooltip');
    var result = listElement.ej2_instances[0].dataSource;
    var i;
    for (i = 0; i < result.length; i++) {

```



```

        if (result[i].text === args.target.textContent) {
            this.content = result[i].content;
            this.dataBind();
            break;
        }
    }
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 DropDownList</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin:0 auto; width:250px;">
        <br>
        <div id="Tooltip" class="e-prevent-select">
            <input type="text" id="ddltooltip">
        </div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

Value change in ##Platform_Name## Drop down list control

You can check about whether value change happened by manual or programmatic by using [change](#) event argument that argument name is `isInteracted`.

The following example demonstrate, how to check whether value change happened by manual or programmatic.

INDEX.JS

```

var sportsData = [
    { Id: 'game1', Game: 'Badminton' },
    { Id: 'game2', Game: 'Football' },
    { Id: 'game3', Game: 'Tennis' }
];
//initiate the DropDownList
var dropDownListObject = new ej.dropdowns.DropDownList({
    // bind the sports Data to datasource property
    dataSource: sportsData,
    // maps the appropriate column to fields property
    fields: { text: 'Game', value: 'Id' },
    //set the placeholder to DropDownList input
    placeholder: "Select a game",
    // bind change event handler
    change: onChange
});
//render the component
dropDownListObject.appendTo('#ddlelement');

// Set value dynamically
document.getElementById('btn').onclick = () => {
    dropDownListObject.value = 'game3';
};
function onChange(args) {
    var element = document.createElement('p');
    if (args.isInteracted) {
        element.innerText = 'Changes happened by Interaction';
    }
    else {
        element.innerText = 'Changes happened by programmatic';
    }
    document.getElementById('event').append(element);
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 DropDownList</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>

```

```

<body>

  <div id="container" style="margin:0 auto; width:250px;">
    <br>
    <input type="text" id="ddlelement">
  </div>
  <button id="btn" class="e-control e-btn"> Set value dynamically
</button>
  <p id="event"> </p>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Value support in ##Platform_Name## Drop down list control

yes, value for each list items should be unique.

Achieve virtual scrolling in ##Platform_Name## Drop down list control

The Virtual Scrolling is used to display a large amount of data without buffering the entire load of a huge database record in the DropDownList, that is, when scrolling, the request is sent and fetch some amount of data from the server dynamically. Using the `scroll` event, get the data and generate the list add to popup using the `addItem` method.

Refer to the following code sample for virtual scrolling.

INDEX.JS

```

var data = new ej.data.DataManager({
  url: 'https://js.syncfusion.com/demos/ejServices/Wcf/Northwind.svc/',
  crossDomain: true
});
//initiates the component
let ddlObject = new ej.dropdowns.DropDownList({
  // bind the DataManager instance to dataSource property
  dataSource: data,
  // bind the Query instance to query property
  query: new
ej.data.Query().from('Customers').select('ContactName').take(7),
  // map the appropriate columns to fields property
  fields: { text: 'ContactName', value: 'ContactName' },
  // set the placeholder to DropDownList input element
  placeholder: 'Select a customer',
  // sort the resulted items
  sortOrder: 'Ascending',
  // set the height of the popup element
  popupHeight: '200px',

  actionComplete: function (e) {
    let operator = new
ej.data.Query().from('Customers').select('ContactName');
    let start = 7;
    let end = 12;

```

```

        let listElement = this.list;
        listElement.addEventListener('scroll', () => {
            if ((listElement.scrollTop + listElement.offsetHeight >=
listElement.scrollHeight)) {
                let filterQuery = operator.clone();
                data.executeQuery(filterQuery.range(start,
end)).then((event) => {
                    start = end;
                    end += 5;
                    ddlObject.addItem(event.result);
                }).catch((e) => {
                });
            }
        })
    });
    ddlObject.appendTo('#atcelement');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 AutoComplete</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin:0 auto; width:250px;">
        <input type="text" tabindex="1" id="atcelement">
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

Dropdown Tree

Data binding in ##Platform_Name## Drop down tree control

The Dropdown Tree control provides an option to load the data either from local data sources or from remote data services. This can be done through `dataSource` property that is a member of the `fields` property. The `dataSource` property supports array of JavaScript objects and `DataManager`. It also supports different kinds of data services such as OData, OData V4, Web API, URL, and JSON with the help of `DataManager` adaptors.

Dropdown Tree has `load on demand` (Lazy load) option. It reduces the bandwidth size when consuming the huge data. By default, the `loadOnDemand` is set to false. By enabling this property, it loads first level items initially, and when parent item is expanded, loads the child items based on the `parentValue/child` member.

Local data

To bind local data to the Dropdown Tree, you can assign a JavaScript object array to the `dataSource` property.

The Dropdown Tree control requires three fields (Value, text, and `parentValue`) to render local data source. When mapper fields are not specified, it takes the default values as the mapping fields. Local data source can also be provided as an instance of the `DataManager`. It supports two kinds of local data binding methods.

- Hierarchical data
- Self-referential data

Hierarchical data

Dropdown Tree can be populated with the hierarchical data source that contains nested array of JSON objects. You can directly map the hierarchical data and the field members with corresponding key values from the hierarchical data to the `fields` property.

In the following example, **code**, **name**, and **countries** columns from the hierarchical data have been mapped to **value**, **text**, and **child** fields, respectively.

INDEX.JS

```
ej.base.enableRipple(true);
//define the nested array of JSON objects
var continents = [
  {
    code: 'AF', name: 'Africa', countries: [
      { code: 'NGA', name: 'Nigeria' },
      { code: 'EGY', name: 'Egypt' },
      { code: 'ZAF', name: 'South Africa' }
    ]
  },
  {
    code: 'AS', name: 'Asia', expanded: true, countries: [
      { code: 'CHN', name: 'China' },
      { code: 'IND', name: 'India', selected: true },
      { code: 'JPN', name: 'Japan' }
    ]
  }
],
```

```

{
  code: 'EU', name: 'Europe', countries: [
    { code: 'DNK', name: 'Denmark' },
    { code: 'FIN', name: 'Finland' },
    { code: 'AUT', name: 'Austria' }
  ]
},
{
  code: 'NA', name: 'North America', countries: [
    { code: 'USA', name: 'United States of America' },
    { code: 'CUB', name: 'Cuba' },
    { code: 'MEX', name: 'Mexico' }
  ]
},
{
  code: 'SA', name: 'South America', countries: [
    { code: 'BRA', name: 'Brazil' },
    { code: 'COL', name: 'Colombia' },
    { code: 'ARG', name: 'Argentina' }
  ]
},
{
  code: 'OC', name: 'Oceania', countries: [
    { code: 'AUS', name: 'Australia' },
    { code: 'NZL', name: 'New Zealand' },
    { code: 'WSM', name: 'Samoa' }
  ]
},
{
  code: 'AN', name: 'Antarctica', countries: [
    { code: 'BVT', name: 'Bouvet Island' },
    { code: 'ATF', name: 'French Southern Lands' }
  ]
},
];
var DropDownTreeObj = new ej.dropdowns.DropDownTree({
  fields: { dataSource: continents, value: 'code', text: 'name', child:
'countries' }
});
DropDownTreeObj.appendTo('#ddltreeelement');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DropDownList</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin:0 auto; width:250px;">
        <input type="text" tabindex="1" id="ddltreeelement">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Self-referential data

Dropdown Tree can be populated from the self-referential data structure that contains array of JSON objects with `parentValue` mapping.

You can directly assign the self-referential data and map all the field members with corresponding key values from self-referential data to the `fields` property.

To render the root level items, specify the `parentValue` as null or no need to specify the `parentValue` in the `dataSource`.

In the following example, `id`, `pid`, `hasChild`, and `name` columns from self-referential data have been mapped to `value`, `parentValue`, `hasChildren`, and `text` fields, respectively.

INDEX.JS

```

ej.base.enableRipple(true);
var localData = [
    { id: 1, name: 'Discover Music', hasChild: true, expanded: true },
    { id: 2, pid: 1, name: 'Hot Singles' },
    { id: 3, pid: 1, name: 'Rising Artists' },
    { id: 4, pid: 1, name: 'Live Music' },
    { id: 6, pid: 1, name: 'Best of 2017 So Far' },
    { id: 7, name: 'Sales and Events', hasChild: true },
    { id: 8, pid: 7, name: '100 Albums - $5 Each' },
    { id: 9, pid: 7, name: 'Hip-Hop and R&B Sale' },
    { id: 10, pid: 7, name: 'CD Deals' },
    { id: 11, name: 'Categories', hasChild: true },
    { id: 12, pid: 11, name: 'Songs' },
    { id: 13, pid: 11, name: 'Bestselling Albums' },

```

```

    { id: 14, pid: 11, name: 'New Releases' },
    { id: 15, pid: 11, name: 'Bestselling Songs' },
    { id: 16, name: 'MP3 Albums', hasChild: true },
    { id: 17, pid: 16, name: 'Rock' },
    { id: 18, pid: 16, name: 'Gospel' },
    { id: 19, pid: 16, name: 'Latin Music' },
    { id: 20, pid: 16, name: 'Jazz' },
    { id: 21, name: 'More in Music', hasChild: true },
    { id: 22, pid: 21, name: 'Music Trade-In' },
    { id: 23, pid: 21, name: 'Redeem a Gift Card' },
    { id: 24, pid: 21, name: 'Band T-Shirts' },
  ];
  var DropDownTreeObj = new ej.dropdowns.DropDownTree({
    fields: { dataSource: localData, value: 'id', parentValue: 'pid', text:
    'name', hasChildren: 'hasChild' }
  });
  DropDownTreeObj.appendTo('#ddltreeelement');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DropDownList</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:0 auto; width:250px;">
    <input type="text" tabindex="1" id="ddltreeelement">
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}

```



```
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Remote data

Dropdown Tree can also be populated from a remote data service with the help of the **DataManager** control and **Query** property.

It supports different kinds of data services such as OData, OData V4, Web API, URL, and JSON with the help of **DataManager** adaptors.

You can assign service data as an instance of **DataManager** to the **dataSource**. To interact with remote data source, you must provide the endpoint url.

The **DataManager** that acts as an interface between the service endpoint and the Dropdown Tree requires the following information to interact with service endpoint properly.

- **DataManager->url**: Defines the service endpoint to fetch data.
- **DataManager->adaptor**: Defines the adaptor option. By default, **ODataAdaptor** is used for remote binding.

Adaptor is responsible for processing response and request from/to the service endpoint. The **@syncfusion/ej2-data** package provides some pre-defined adaptors designed to interact with service endpoints. They are,

- **UrlAdaptor**: Used to interact with remote services. This is the base adaptor for all remote based adaptors.
- **ODataAdaptor**: Used to interact with OData endpoints.
- **ODataV4Adaptor**: Used to interact with OData V4 endpoints.
- **WebApiAdaptor**: Used to interact with Web API created under OData standards.
- **WebMethodAdaptor**: Used to interact with web methods.

In the following example, **ODataV4Adaptor** is used to fetch data from the remote services. The **EmployeeID**, **FirstName**, and **EmployeeID**

columns from the Employees table have been mapped to **value**, **text**, and **hasChildren** fields respectively for first level items.

The **OrderID**, **EmployeeID**, and **ShipName** columns from the orders table have been mapped to **value**, **parentValue**, and **text** fields respectively for second level items.

INDEX.JS

```
ej.base.enableRipple(true);
var data = new ej.data.DataManager({
  url: 'https://services.odata.org/V4/Northwind/Northwind.svc',
  adaptor: new ej.data.ODataV4Adaptor(),
  crossDomain: true,
});
```

```

var query = new
ej.data.Query().from('Employees').select('EmployeeID,FirstName,Title').take(
5);
var query1 = new
ej.data.Query().from('Orders').select('OrderID,EmployeeID,ShipName').take(5)
;
var DropDownTreeObj = new ej.dropdowns.DropDownTree({
    fields: { dataSource: data, query: query, value: 'EmployeeID', text:
'FirstName', hasChildren: 'EmployeeID', tooltip: 'Title',
        child: { dataSource: data, query: query1, value: 'OrderID',
parentValue: 'EmployeeID', text: 'ShipName' }
    }
});
DropDownTreeObj.appendTo('#ddltreeelement');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 DropDownList</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin:0 auto; width:250px;">
        <input type="text" tabindex="1" id="ddltreeelement">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Prevent Node selection

You can prevent the selection of individual tree node by using the [selectable](#) property. The tree node selection is not allowed while disable this property.

The `selectable` property is disabled and the selection is prevented for parent nodes in below sample.

INDEX.JS

```
ej.base.enableRipple(true);
let localData = [
  { id: 1, name: 'Discover Music', hasChild: true, expanded: true,
selectable: false },
  { id: 2, pid: 1, name: 'Hot Singles' },
  { id: 3, pid: 1, name: 'Rising Artists' },
  { id: 4, pid: 1, name: 'Live Music' },
  { id: 6, pid: 1, name: 'Best of 2017 So Far' },
  { id: 7, name: 'Sales and Events', hasChild: true, selectable: false },
  { id: 8, pid: 7, name: '100 Albums' },
  { id: 9, pid: 7, name: 'Hip-Hop and R&B Sale' },
  { id: 10, pid: 7, name: 'CD Deals' },
  { id: 11, name: 'Categories', hasChild: true, selectable: false },
  { id: 12, pid: 11, name: 'Songs' },
  { id: 13, pid: 11, name: 'Bestselling Albums' },
  { id: 14, pid: 11, name: 'New Releases' },
  { id: 15, pid: 11, name: 'Bestselling Songs' },
  { id: 16, name: 'MP3 Albums', hasChild: true, selectable: false },
  { id: 17, pid: 16, name: 'Rock' },
  { id: 18, pid: 16, name: 'Gospel' },
  { id: 19, pid: 16, name: 'Latin Music' },
  { id: 20, pid: 16, name: 'Jazz' },
  { id: 21, name: 'More in Music', hasChild: true, selectable: false },
  { id: 22, pid: 21, name: 'Music Trade-In' },
  { id: 23, pid: 21, name: 'Redeem a Gift Card' },
  { id: 24, pid: 21, name: 'Band T-Shirts' },
];
var DropDownTreeObj = new ej.dropdowns.DropDownTree({
  fields: { dataSource: localData, value: 'id', parentValue: 'pid', text:
'name', hasChildren: 'hasChild', selectable: 'selectable' }
});
DropDownTreeObj.appendTo('#ddTree');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DropDownTree</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin:0 auto; width:250px;">
        <input type="text" tabindex="1" id="ddTree">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Templates in ##Platform_Name## Drop down tree control

The Dropdown Tree provides support to customize each list item, header, and footer elements. It uses the Essential JS 2 [Template engine](#) to compile and render the elements properly.

Item template

The content of each list item within the Dropdown Tree can be customized with the help of [itemTemplate](#) property.

In the following sample, the Dropdown Tree list items are customized with employee information such as **name** and **job** using the **itemTemplate** property.

The template expression should be provided inside the `${...}` interpolation syntax.

INDEX.JS

```

var data = [
    { "id": 1, "name": "Steven Buchanan", "job": "General Manager",
      "hasChild": true, "expanded": true },
    { "id": 2, "pid": 1, "name": "Laura Callahan", "job": "Product Manager",
      "hasChild": true },
    { "id": 3, "pid": 2, "name": "Andrew Fuller", "job": "Team Lead",
      "hasChild": true },
    { "id": 4, "pid": 3, "name": "Anne Dodsworth", "job": "Developer" },
    { "id": 10, "pid": 3, "name": "Lilly", "job": "Developer", "status":
      "online" },
    { "id": 5, "pid": 1, "name": "Nancy Davolio", "job": "Product Manager",
      "hasChild": true },

```

```

    { "id": 6, "pid": 5, "name": "Michael Suyama", "job": "Team Lead",
      "hasChild": true },
    { "id": 7, "pid": 6, "name": "Robert King", "job": "Developer " },
    { "id": 11, "pid": 6, "name": "Mary", "job": "Developer " },
    { "id": 9, "pid": 1, "name": "Janet Leverling", "job": "HR" }
  ];
  var ddtreeObj = new ej.dropdowns.DropDownTree({
    fields: { dataSource: data, text: 'name', value: 'id', parentValue:
      'pid', hasChildren: 'hasChild' },
    itemTemplate: '#itemTemplate',
    width: '100%',
    cssClass: 'custom',
    placeholder: 'Select an employee',
    popupHeight: '250px'
  });
  ddtreeObj.appendTo('#template');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DropDownTree</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div class="stackblitz-container bootstrap5">
    <div class="col-lg-12 control-section">
      <div style="margin: 0 auto; max-width:350px;">
        <input type="text" id="template">
      </div>
    </div>
    <script id="itemTemplate" type="text/x-template">
      <div>
        <span class="ename">${name} - </span>
        <span class="ejob">${job}</span>
      </div>
    </script>
  </div>

```

```

        </div>
    </script>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Header template

The header element is shown statically at the top of the popup list items within the Dropdown Tree. A custom element can be placed as a header element using the [headerTemplate](#) property.

In the following sample, the header is customized with the custom element.

INDEX.JS

```

var data = [
    { "id": 1, "name": "Steven Buchanan", "job": "General Manager",
    "hasChild": true, "expanded": true },
    { "id": 2, "pid": 1, "name": "Laura Callahan", "job": "Product Manager",
    "hasChild": true },
    { "id": 3, "pid": 2, "name": "Andrew Fuller", "job": "Team Lead",
    "hasChild": true },
    { "id": 4, "pid": 3, "name": "Anne Dodsworth", "job": "Developer" },
    { "id": 10, "pid": 3, "name": "Lilly", "job": "Developer", "status":
    "online" },
    { "id": 5, "pid": 1, "name": "Nancy Davolio", "job": "Product Manager",
    "hasChild": true },
    { "id": 6, "pid": 5, "name": "Michael Suyama", "job": "Team Lead",
    "hasChild": true },
    { "id": 7, "pid": 6, "name": "Robert King", "job": "Developer " },
    { "id": 11, "pid": 6, "name": "Mary", "job": "Developer " },
    { "id": 9, "pid": 1, "name": "Janet Leverling", "job": "HR" }
];
var ddtreeObj = new ej.dropdowns.DropDownTree({
    fields: { dataSource: data, text: 'name', value: 'id', parentValue:
    'pid', hasChildren: 'hasChild' },
    headerTemplate: '#headerTemplate',
    width: '100%',
    cssClass: 'custom',
    placeholder: 'Select an employee',
    popupHeight: '250px'
});
ddtreeObj.appendTo('#template');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 DropDownTree</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">

```

```

<meta name="author" content="Syncfusion">
<link href="styles.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div class="stackblitz-container bootstrap5">
    <div class="col-lg-12 control-section">
      <div style="margin: 0 auto; max-width:350px;">
        <input type="text" id="template">
      </div>
    </div>
    <script id="headerTemplate" type="text/x-template">
      <div class="head"> Employee List </div>
    </script>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Footer template

The Dropdown Tree has options to show a footer element at the bottom of the list items in the popup list. Here, you can place any custom element as a footer element using the [footerTemplate](#) property.

In the following sample, the footer element displays the total number of employees present in the Dropdown Tree.

INDEX.JS

```

var data = [
  { "id": 1, "name": "Steven Buchanan", "job": "General Manager",
    "hasChild": true, "expanded": true },
  { "id": 2, "pid": 1, "name": "Laura Callahan", "job": "Product Manager",
    "hasChild": true },

```

```

    { "id": 3, "pid": 2, "name": "Andrew Fuller", "job": "Team Lead",
      "hasChild": true },
    { "id": 4, "pid": 3, "name": "Anne Dodsworth", "job": "Developer" },
    { "id": 10, "pid": 3, "name": "Lilly", "job": "Developer", "status":
      "online" },
    { "id": 5, "pid": 1, "name": "Nancy Davolio", "job": "Product Manager",
      "hasChild": true },
    { "id": 6, "pid": 5, "name": "Michael Suyama", "job": "Team Lead",
      "hasChild": true },
    { "id": 7, "pid": 6, "name": "Robert King", "job": "Developer " },
    { "id": 11, "pid": 6, "name": "Mary", "job": "Developer " },
    { "id": 9, "pid": 1, "name": "Janet Leverling", "job": "HR" }
  ];
  var ddtreeObj = new ej.dropdowns.DropDownTree({
    fields: { dataSource: data, text: 'name', value: 'id', parentValue:
      'pid', hasChildren: 'hasChild' },
    footerTemplate: "<span class='foot'> Total number of employees: " +
      data.length + "</span>",
    width: '100%',
    cssClass: 'custom',
    placeholder: 'Select an employee',
    popupHeight: '250px'
  });
  ddtreeObj.appendTo('#template');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DropDownTree</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>
  <div class="stackblitz-container bootstrap5">
    <div class="col-lg-12 control-section">

```



```

        <div style="margin: 0 auto; max-width:350px;">
            <input type="text" id="template">
        </div>
    </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

No records template

The Dropdown Tree is supports to display custom design in the popup list content using the [noRecordsTemplate](#) property when no matches found on search.

In the following sample, popup list content displays the notification of no data available.

INDEX.JS

```

var data = [];
var ddtreeObj = new ej.dropdowns.DropDownTree({
    fields: { dataSource: data, text: 'name', value: 'id', parentValue:
'pid', hasChildren: 'hasChild' },
    noRecordsTemplate: "<span class='norecord'> NO DATA AVAILABLE</span>",
    width: '100%',
    placeholder: 'Select an employee',
    popupHeight: '250px'
});
ddtreeObj.appendTo('#template');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 DropDownTree</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <div class="stackblitz-container bootstrap5">
        <div class="col-lg-12 control-section">
            <div style="margin: 0 auto; max-width:350px;">
                <input type="text" id="template">
            </div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Action failure template

The Dropdown Tree provides an option to custom design the popup list content using [actionFailureTemplate](#) property, when the data fetch request fails at the remote server.

In the following sample, when the data fetch request fails, the Dropdown Tree displays the notification.

INDEX.JS

```

var data = new ej.data.DataManager({
    url: 'https://services.odata.org/V4/Northwind/Northwind.svs',
    adaptor: new ej.data.ODataV4Adaptor,
    crossDomain: true,
});
var query = new
ej.data.Query().from('Employees').select('EmployeeID,FirstName,Title').take(
5);
var query1 = new
ej.data.Query().from('Orders').select('OrderID,EmployeeID,ShipName').take(5)
;
var ddTreeObj = new ej.dropdowns.DropDownTree({
    fields: {
        dataSource: data, query: query, value: 'EmployeeID', text:
'FirstName', hasChildren: 'EmployeeID',
        child: { dataSource: data, query: query1, value: 'OrderID',
parentValue: 'EmployeeID', text: 'ShipName' }
    },
    width: '100%',
    actionFailureTemplate: "<span class='action-failure'> Data fetch request
fails</span>",
    placeholder: 'Select an employee',
    popupHeight: '250px'
});
ddTreeObj.appendTo('#template');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DropDownTree</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div class="stackblitz-container bootstrap5">
    <div class="col-lg-12 control-section">
      <div style="margin: 0 auto; max-width:350px;">
        <input type="text" id="template">
      </div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Custom template to show selected items in input

In Dropdown Tree, while selecting more than one items via checkbox or multi selection support, all the selected items will be displayed in the input. Instead of displaying all the selected item text, the custom template can be displayed by setting the the [mode](#) property as **Custom** and [customTemplate](#) property.

When the **mode** property is set as **Custom**, the Dropdown Tree displays the default template value **(\${value.length} item(s) selected)** like **1 item(s) selected** or **2 item(s) selected**. The default template can be customized by setting **customTemplate** property.

In the following sample, the Dropdown Tree is rendered with default value of the **customTemplate** property like “1 item(s) selected or 2 item(s) selected”.

INDEX.JS

```
var data = [
  { "id": 1, "name": "Steven Buchanan", "job": "General Manager",
    "hasChild": true, "expanded": true },
  { "id": 2, "pid": 1, "name": "Laura Callahan", "job": "Product Manager",
    "hasChild": true },
  { "id": 3, "pid": 2, "name": "Andrew Fuller", "job": "Team Lead",
    "hasChild": true },
  { "id": 4, "pid": 3, "name": "Anne Dodsworth", "job": "Developer" },
  { "id": 10, "pid": 3, "name": "Lilly", "job": "Developer", "status":
    "online" },
  { "id": 5, "pid": 1, "name": "Nancy Davolio", "job": "Product Manager",
    "hasChild": true },
  { "id": 6, "pid": 5, "name": "Michael Suyama", "job": "Team Lead",
    "hasChild": true },
  { "id": 7, "pid": 6, "name": "Robert King", "job": "Developer" },
  { "id": 11, "pid": 6, "name": "Mary", "job": "Developer" },
  { "id": 9, "pid": 1, "name": "Janet Leverling", "job": "HR" }
];
var checkList = new ej.dropdowns.DropDownTree({
  fields: { dataSource: data, text: 'name', value: 'id', parentValue:
    'pid', hasChildren: 'hasChild' },
  placeholder: 'Select items',
  popupHeight: '200px',
  mode: 'Custom',
  showCheckBox: true,
  treeSettings: { autoCheck: true }
});
checkList.appendTo('#checkbox');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DropDownTree</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <div class="stackblitz-container bootstrap5">
        <div class="col-lg-12 control-section">
            <div style="margin: 0 auto; max-width:350px;">
                <input type="text" id="checkbox">
            </div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

In the following sample, the Dropdown Tree is rendered with custom value of the **customTemplate** property like **Selected items count: 2**.

INDEX.JS

```

var data = [
    { "id": 1, "name": "Steven Buchanan", "job": "General Manager",
    "hasChild": true, "expanded": true },
    { "id": 2, "pid": 1, "name": "Laura Callahan", "job": "Product Manager",
    "hasChild": true },
    { "id": 3, "pid": 2, "name": "Andrew Fuller", "job": "Team Lead",
    "hasChild": true },
    { "id": 4, "pid": 3, "name": "Anne Dodsworth", "job": "Developer" },
    { "id": 10, "pid": 3, "name": "Lilly", "job": "Developer", "status":
    "online" },
    { "id": 5, "pid": 1, "name": "Nancy Davolio", "job": "Product Manager",
    "hasChild": true },
    { "id": 6, "pid": 5, "name": "Michael Suyama", "job": "Team Lead",
    "hasChild": true },
    { "id": 7, "pid": 6, "name": "Robert King", "job": "Developer " },
    { "id": 11, "pid": 6, "name": "Mary", "job": "Developer " },
    { "id": 9, "pid": 1, "name": "Janet Leverling", "job": "HR" }
];
var checkList = new ej.dropdowns.DropDownTree({
    fields: { dataSource: data, text: 'name', value: 'id', parentValue:
    'pid', hasChildren: 'hasChild' },
    placeholder: 'Select items',
    popupHeight: '200px',
    mode: 'Custom',
    customTemplate: "Selected items count: ${value.length}",
    showCheckBox: true,
    treeSettings: { autoCheck: true }
});

```

```
checkList.appendTo('#checkbox');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DropDownTree</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div class="stackblitz-container bootstrap5">
    <div class="col-lg-12 control-section">
      <div style="margin: 0 auto; max-width:350px;">
        <input type="text" id="checkbox">
      </div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Checkbox in ##Platform_Name## Drop down tree control

The Dropdown Tree control allows you to check more than one item from the tree without affecting the UI's appearance by enabling the `showCheckBox` property. When this property is enabled, checkbox appears before each item text in the popup.

In the following example, the `showCheckBox` property is enabled.

INDEX.JS

```

var localData = [
  { id: 1, name: 'Discover Music', hasChild: true, expanded: true },
  { id: 2, pid: 1, name: 'Hot Singles' },
  { id: 3, pid: 1, name: 'Rising Artists' },
  { id: 4, pid: 1, name: 'Live Music' },
  { id: 6, pid: 1, name: 'Best of 2017 So Far' },
  { id: 7, name: 'Sales and Events', hasChild: true },
  { id: 8, pid: 7, name: '100 Albums - $5 Each' },
  { id: 9, pid: 7, name: 'Hip-Hop and R&B Sale' },
  { id: 10, pid: 7, name: 'CD Deals' },
  { id: 11, name: 'Categories', hasChild: true },
  { id: 12, pid: 11, name: 'Songs' },
  { id: 13, pid: 11, name: 'Bestselling Albums' },
  { id: 14, pid: 11, name: 'New Releases' },
  { id: 15, pid: 11, name: 'Bestselling Songs' },
  { id: 16, name: 'MP3 Albums', hasChild: true },
  { id: 17, pid: 16, name: 'Rock' },
  { id: 18, pid: 16, name: 'Gospel' },
  { id: 19, pid: 16, name: 'Latin Music' },
  { id: 20, pid: 16, name: 'Jazz' },
  { id: 21, name: 'More in Music', hasChild: true },
  { id: 22, pid: 21, name: 'Music Trade-In' },
  { id: 23, pid: 21, name: 'Redeem a Gift Card' },
  { id: 24, pid: 21, name: 'Band T-Shirts' },
];
var DropDownTreeObj = new ej.dropdowns.DropDownTree({
  fields: { dataSource: localData, value: 'id', parentValue: 'pid', text:
'name', hasChildren: 'hasChild' },
  showCheckBox: true
});
DropDownTreeObj.appendTo('#ddltreeelement');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DropDownList</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin:0 auto; width:250px;">
        <input type="text" tabindex="1" id="ddltreeelement">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Auto Check

By default, the checkbox state of the parent and child items in the Dropdown Tree will not be dependent over each other. If you need dependent checked state, then enable the `autoCheck` property which is a member of `treeSettings` property.

- If one or more child items are not in the checked state, then the parent item will be in the intermediate state.
- If all the child items are checked, then the parent item will also be in the checked state.
- If a parent item is checked, then all the child items will also be changed to the checked state.

In the following example, the `autoCheck` property is enabled.

INDEX.JS

```

var localData = [
    { id: 1, name: 'Discover Music', hasChild: true, expanded: true },
    { id: 2, pid: 1, name: 'Hot Singles' },
    { id: 3, pid: 1, name: 'Rising Artists' },
    { id: 4, pid: 1, name: 'Live Music' },
    { id: 5, pid: 1, name: 'Best of 2017 So Far' },
    { id: 7, name: 'Sales and Events', hasChild: true },
    { id: 8, pid: 7, name: '100 Albums - $5 Each' },
    { id: 9, pid: 7, name: 'Hip-Hop and R&B Sale' },
    { id: 10, pid: 7, name: 'CD Deals' },
    { id: 11, name: 'Categories', hasChild: true },
    { id: 12, pid: 11, name: 'Songs' },
    { id: 13, pid: 11, name: 'Bestselling Albums' },
    { id: 14, pid: 11, name: 'New Releases' },
    { id: 15, pid: 11, name: 'Bestselling Songs' },
    { id: 16, name: 'MP3 Albums', hasChild: true },
    { id: 17, pid: 16, name: 'Rock' },
    { id: 18, pid: 16, name: 'Gospel' },
    { id: 19, pid: 16, name: 'Latin Music' },
    { id: 20, pid: 16, name: 'Jazz' },
    { id: 21, name: 'More in Music', hasChild: true },

```



```

    { id: 22, pid: 21, name: 'Music Trade-In' },
    { id: 23, pid: 21, name: 'Redeem a Gift Card' },
    { id: 24, pid: 21, name: 'Band T-Shirts' },
  ];
  var DropDownTreeObj = new ej.dropdowns.DropDownTree({
    fields: { dataSource: localData, value: 'id', parentValue: 'pid', text:
    'name', hasChildren: 'hasChild' },
    showCheckBox: true,
    treeSettings: { autoCheck: true }
  });
  DropDownTreeObj.appendTo('#ddltreeelement');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DropDownList</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:0 auto; width:250px;">
    <input type="text" tabindex="1" id="ddltreeelement">
  </div>
  <script>
  var ele = document.getElementById('container');
  if(ele) {
    ele.style.visibility = "visible";
  }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Select All

The Dropdown Tree control has in-built support to select all the tree items using Select All options in the header.

When the `showSelectAll` property is set to true, a checkbox will be displayed in the popup header that allows you to select or deselect all the tree items in the popup.

By default, `Select All` and `unSelect All` text values will be showcased along with the checkbox in the popup header to indicate the action to be performed on checking or unchecking the checkbox. You can customize these name attributes by using `selectAllText` and `unSelectAllText` properties respectively.

INDEX.JS

```
var localData = [
  { id: 1, name: 'Discover Music', hasChild: true, expanded: true },
  { id: 2, pid: 1, name: 'Hot Singles' },
  { id: 3, pid: 1, name: 'Rising Artists' },
  { id: 4, pid: 1, name: 'Live Music' },
  { id: 6, pid: 1, name: 'Best of 2017 So Far' },
  { id: 7, name: 'Sales and Events', hasChild: true },
  { id: 8, pid: 7, name: '100 Albums - $5 Each' },
  { id: 9, pid: 7, name: 'Hip-Hop and R&B Sale' },
  { id: 10, pid: 7, name: 'CD Deals' },
  { id: 11, name: 'Categories', hasChild: true },
  { id: 12, pid: 11, name: 'Songs' },
  { id: 13, pid: 11, name: 'Bestselling Albums' },
  { id: 14, pid: 11, name: 'New Releases' },
  { id: 15, pid: 11, name: 'Bestselling Songs' },
  { id: 16, name: 'MP3 Albums', hasChild: true },
  { id: 17, pid: 16, name: 'Rock' },
  { id: 18, pid: 16, name: 'Gospel' },
  { id: 19, pid: 16, name: 'Latin Music' },
  { id: 20, pid: 16, name: 'Jazz' },
  { id: 21, name: 'More in Music', hasChild: true },
  { id: 22, pid: 21, name: 'Music Trade-In' },
  { id: 23, pid: 21, name: 'Redeem a Gift Card' },
  { id: 24, pid: 21, name: 'Band T-Shirts' },
];

var DropDownTreeObj = new ej.dropdowns.DropDownTree({
  fields: { dataSource: localData, value: 'id', parentValue: 'pid', text:
'name', hasChildren: 'hasChild' },
  showCheckBox: true,
  showSelectAll: true,
  selectAllText: 'Check All',
  unSelectAllText: 'UnCheck All'
});

DropDownTreeObj.appendTo('#ddltreeelement');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DropDownList</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
```

```

<link href="styles.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin:0 auto; width:250px;">
        <input type="text" tabindex="1" id="ddltreeelement">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Localization in ##Platform_Name## Drop down tree control

The [Localization](#) library allows you to localize default text content of the Dropdown Tree and it can be localized to any culture by defining the texts and messages of the Dropdown Tree in the corresponding culture. The default locale of the Dropdown Tree is en (English). The following table represents the default texts and messages of the Dropdown Tree in en culture.

|KEY|Text/Message|

|----|----|

|noRecordsTemplate|No records found|

|actionFailureTemplate|Request failed|

|overflowCountTemplate|+\${count} more..|

|totalCountTemplate|\${count} selected|

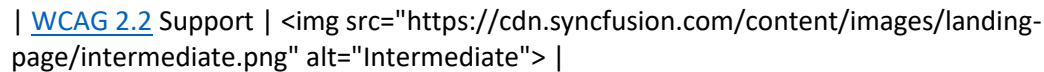
Accessibility in ##Platform_Name## Dropdown Tree component

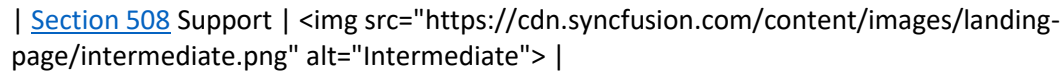
The Dropdown Tree component followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

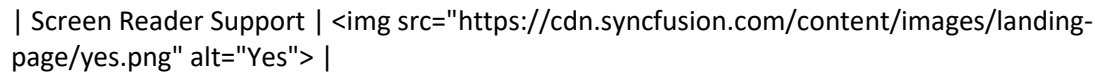
The accessibility compliance for the Dropdown Tree component is outlined below.

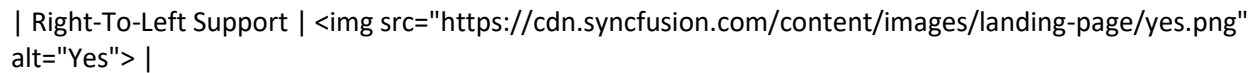
| Accessibility Criteria | Compatibility |

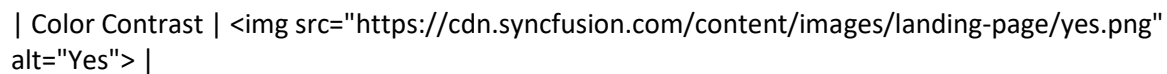
| -- | -- |

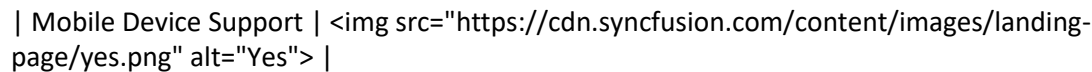
| [WCAG 2.2](#) Support |  |

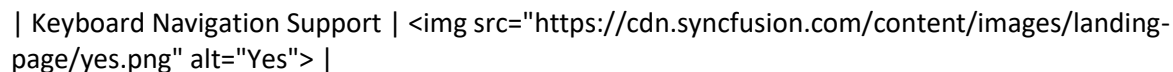
| [Section 508](#) Support |  |

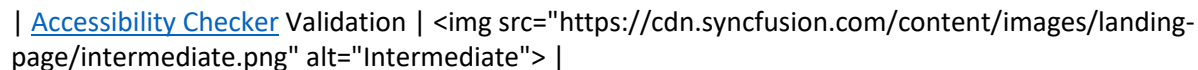
| Screen Reader Support |  |

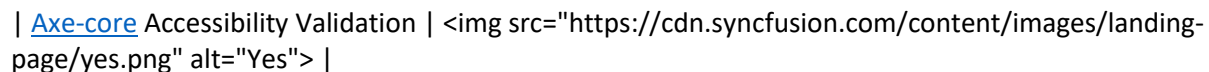
| Right-To-Left Support |  |

| Color Contrast |  |

| Mobile Device Support |  |

| Keyboard Navigation Support |  |

| [Accessibility Checker](#) Validation |  |

| [Axe-core](#) Accessibility Validation |  |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<div> - All features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>

[WAI-ARIA attributes](#)

The Dropdown Tree component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Dropdown Tree component:

| Attributes | Purpose |

--- ---	
role=listbox	All list items are contained within the element.
aria-disabled	Indicates element is perceivable but disabled.
aria-owns	This attribute contains the ID of the popup list to indicate popup as a child element.
aria-haspopup	Indicates whether the Dropdown Tree input element has a popup list or not.
aria-expanded	Indicates the state of the popup list for Dropdown Tree and the parent node's expansion status for TreeView.
aria-activedescendent	This attribute holds the ID of the active list item to focus its descendant child element.
aria-labelledby	This attribute points to the element(s) labeling the element it's applied to.
aria-describedby	This attribute points to the element(s) describing the one it's set on.
role=tree	All tree nodes are contained within the element.
role=treeitem	Specifies the role of each tree node in a selectable TreeView and its containment within the tree.
role=group	Specifies the role of each parent node container in the TreeView.
role=checkbox	Indicates checkbox control along with treeitem element.
aria-multiselectable	Indicates whether the TreeView enables multiple selection or not.
aria-selected	Indicates the selected node.
aria-level	Indicates the level of node in TreeView.
aria-checked	Indicates the current checked state of TreeView checkbox.
aria-label	Indicates the contextual message for the TreeView checkbox and Dropdown Tree.
aria-activedescendant	Identifies the currently active element when focusing on the TreeView.

Keyboard interaction

The Dropdown Tree component followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Dropdown Tree component.

Interaction Keys	Description

Alt + Down	Opens the popup.
Alt + Up	Closes the popup.
Esc(Escape)	Closes the popup when it is in an open state.
Arrow Up	Goes to the previous item in the popup.
Arrow Down	Goes to the next item in the popup.

| Arrow Right | Expands the current item in the popup. |

| Arrow Left | Collapses the current item in the popup. |

| Home | Goes to the first item in the popup. |

| End | Goes to the last item in the popup. |

| Enter | Selects the focused item in the popup. |

| Space | Checks the current item in the popup. |

Ensuring accessibility

The Dropdown Tree component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Dropdown Tree component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Dropdown Tree component with accessibility tools.

See also

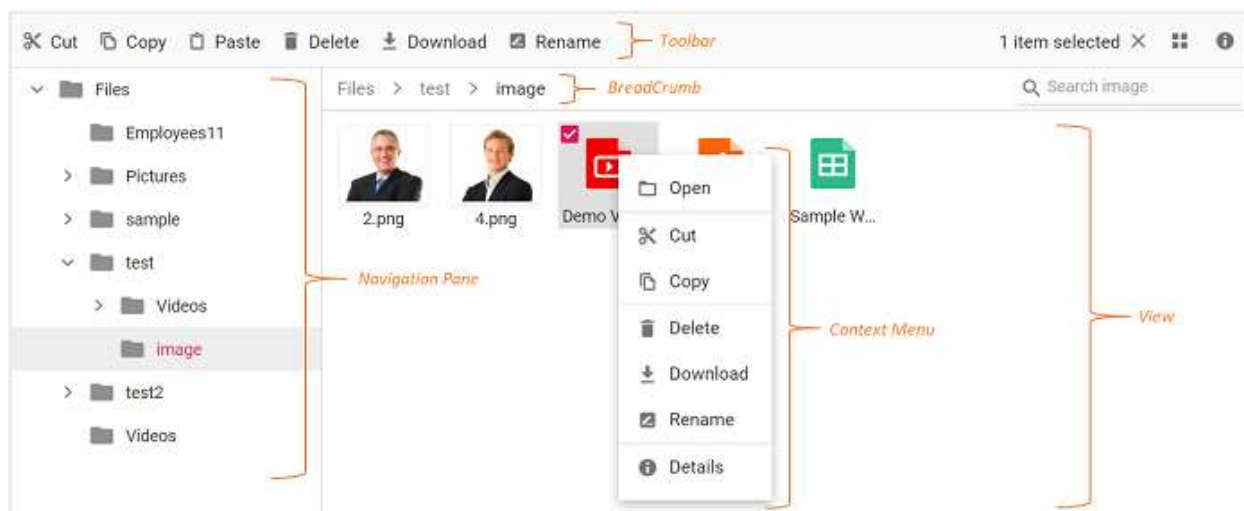
- [Accessibility in Syncfusion ##Platform_Name## components](#)

FileManager

User interface in ##Platform_Name## File manager control

The file manager UI is comprised of several sections like view, toolbar, breadcrumb, context menu, and so on. The UI of the file manager is enhanced with injectable modules like **Details View** for browsing files and folders in a grid, **Navigation Pane** for folder navigation, and **Toolbar** for file operations. The file manager with all feature modules have the following sections in its UI.

- [Toolbar](#) (For direct access to file operations)
- [Navigation Pane](#) (For easy navigation between folders)
- [Breadcrumb](#) (For parent folder navigations)
- [View](#) (For browsing files and folders using large icon view or details view)
- [Context Menu](#) (For accessing file operations)



The basic file manager is a light weight component with all the basic functions. The basic file manager have the following sections in its UI to browse files and folders and manage them with file operations.

- [Breadcrumb](#) (For parent folder navigations)
- [View](#) (Large Icons view for browsing files and folders)
- [Context Menu](#) (For accessing file operations)

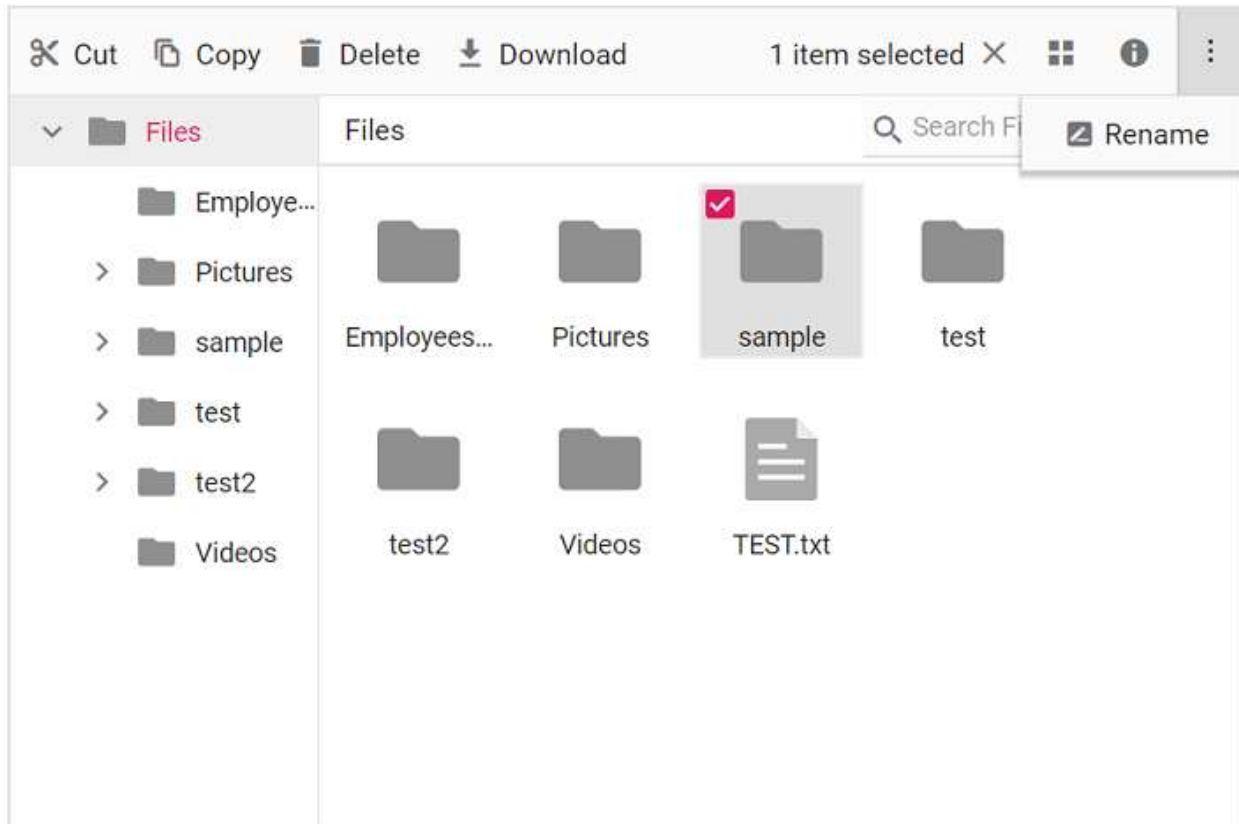


Toolbar

The **Toolbar** provides easy access to the file operations using different buttons and it is presented at the top of the file manager.

If the toolbar items exceed the size of the toolbar, then the exceeding toolbar size will be moved to toolbar popup with a dropdown button at the end of toolbar.

**Refer [Toolbar](#) section in file operations to know more about the buttons present in toolbar*.*



Files and folders navigation

The file manager provides navigation between files and folders using the following two options.

- [Navigation Pane](#)
- [Breadcrumb](#)

Navigation pane

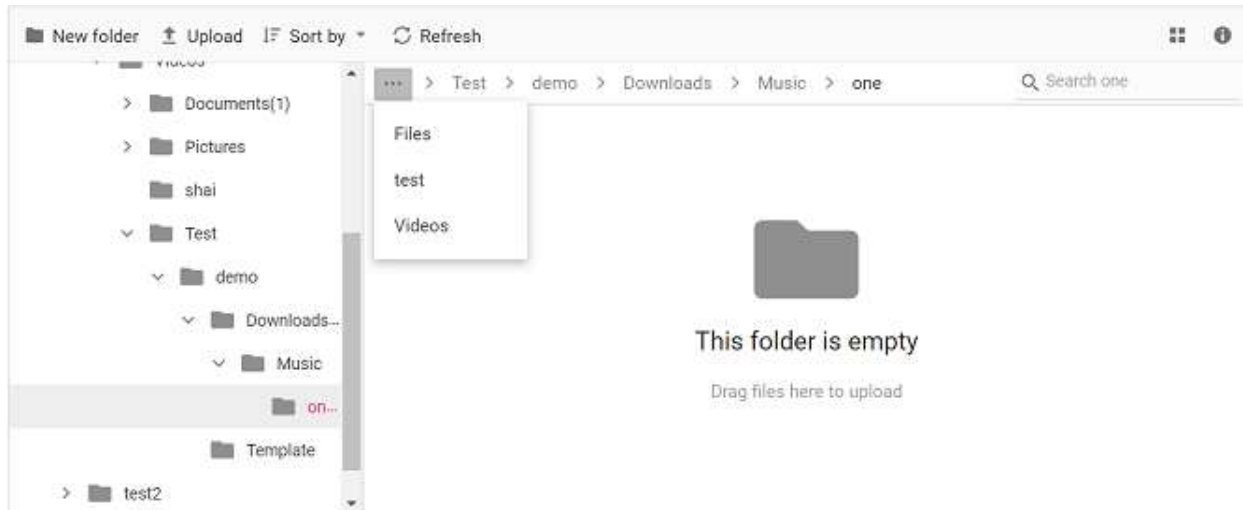
The navigation pane is an injectable module so, it should be injected before rendering the file manager to use its functionality.

It displays the folder hierarchy of the file system and provides easy navigation to the desired folder. Using [navigationPaneSettings](#) minimum and maximum width of the navigation pane can be changed. The navigation pane can be shown or hidden using the `visible` option in the [navigationPaneSettings](#).

BreadCrumb

The file manager provides breadcrumb for navigating to the parent folders. The breadcrumb the in file manager is responsible for resizing.

Whenever the path length exceeds the breadcrumb length, a dropdown button will be added at the starting of the breadcrumb to hold the parent folders adjacent to root.



View

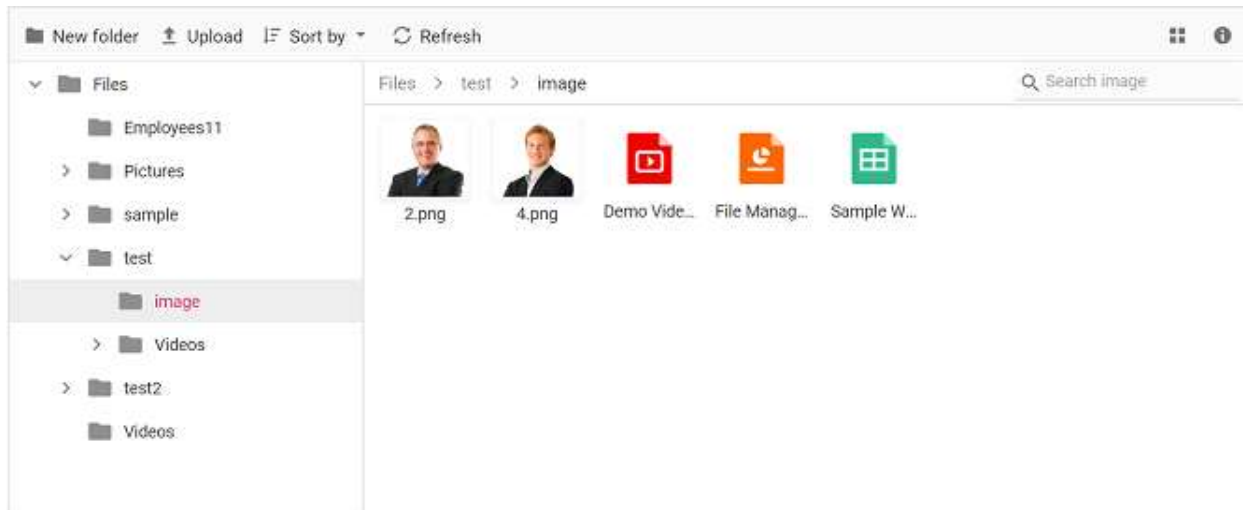
View is the section where the files and folders are displayed for the user to browse. The file manager has two types of views to display the files and folders.

- [Large Icons View](#)
- [Details View](#)

The **large icons view** is the default starting view in the file manager. The view can be changed by using the [toolbar](#) view button or by using the view menu in [context menu](#). The [view](#) API can also be used to change the initial view of the file manager.

Large icons view

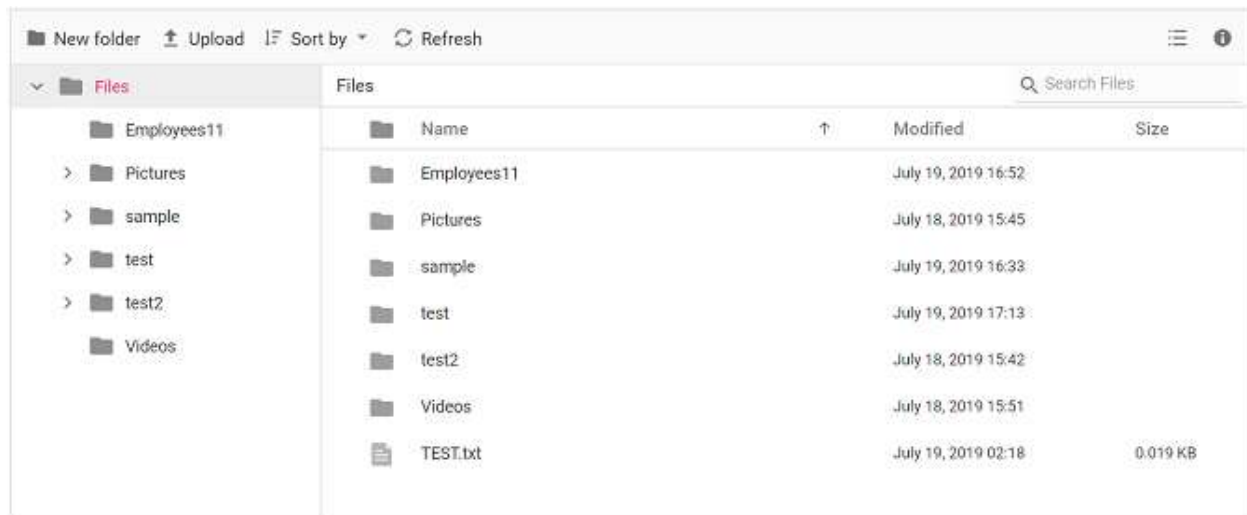
In the large icons view, the thumbnail icons will be shown in a larger size, which displays the data in a form that best suits their content. For image and video type files, a **preview** will be displayed. Extension thumbnails will be displayed for other type files.



Details view

Details view is an injectable module in the file manager so, it should be injected before rendering the file manager to avail its functionality. In the details view, the files are displayed in a sorted list order. This

file list comprises of several columns of information about the files such as **Name**, **Date Modified**, **Type**, and **Size**. Each file has its own small icon representing the file type. Additional columns can be added using [detailsViewSettings](#) API. The details view allows you to perform sorting using column header.

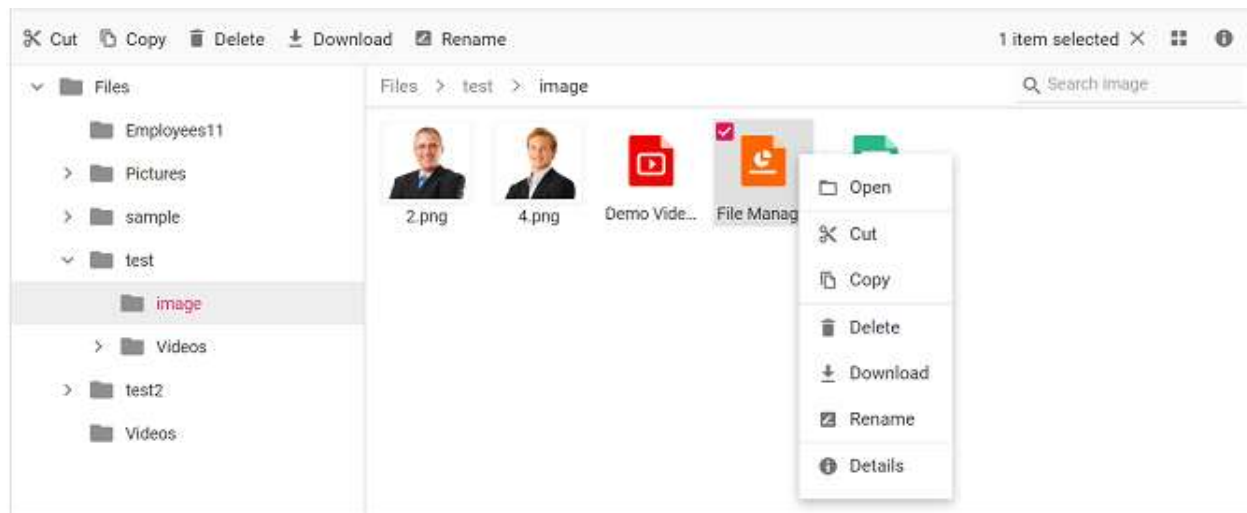


Context menu

The context menu appears on user interaction such as right-click. The file manager is provided with context menu support to perform list of file operations with the files and folders. Context menu appears with varying menu items based on the targets such as file, folder (including navigation pane folders), and layout (empty area in view).

Context menu can be customized using the [contextMenuSettings](#), [menuOpen](#), and [menuClick](#) events.

**Refer [Context Menu](#) section in file operations to know more about the menu items present in context menu*.*



File operations in ##Platform_Name## File manager control

The file manager component is used to browse, manage, and organize the files and folders in a file system through a web application. All basic file operations like creating a new folder, uploading and downloading of files in the file system, and deleting and renaming of existing files and folders are

available in the file manager component. Additionally, previewing of image files is also provided in the file manager component.

The following table represents the basic operations available in the file manager and their corresponding functions.

Operation Name	Function
read	Read the details of files or folders available in the given path from the file system, to display the files for the user to browse the content.
create	Creates a new folder in the current path of the file system.
delete	Removes the file or folder from the file server.
rename	Rename the selected file or folder in the file system.
search	Searches for items matching the search string in the current and child directories.
details	Gets the detail of the selected item(s) from the file server.
copy	Copy the selected file or folder in the file system.
move	Cut the selected file or folder in the file server.
upload	Upload files to the current path or directory in the file system.
download	Downloads the file from the server and the multiple files can be downloaded as ZIP files.

The *CreateFolder*, *Remove*, and *Rename* actions will be reflected in the file manager only after the successful response from the server.

Folder Upload support

To perform the directory(folder) upload in File Manager, set [directoryUpload](#) as true within the uploadSettings property. The directory upload feature is supported for the following file service providers:

- Physical file service provider.
- Azure file service provider.
- NodeJS file service provider.
- Amazon file service provider.

In the following example, directory upload is enabled/disabled on DropDownButton selection.

INDEX.JS

```
var hostUrl = 'https://ej2-aspcore-service.azurewebsites.net/';
// Initialize the FileManager component
var fileObject = new ej.filemanager.FileManager({
    ajaxSettings: {
        url: hostUrl + 'api/FileManager/FileOperations',
        getImageUrl: hostUrl + 'api/FileManager/GetImage',
        uploadUrl: hostUrl + 'api/FileManager/Upload',
        downloadUrl: hostUrl + 'api/FileManager/Download'
    },
});
fileObject.appendTo('#file');
```

```

var items = [{ text: 'Folder' }, { text: 'Files' }];
var drpDownBtn = new ej.splitbuttons.DropDownButton({
    items: items,
    select: function (args) {
        if (args.item.text === 'Folder') {
            fileObject.uploadSettings.directoryUpload = true;
        } else {
            fileObject.uploadSettings.directoryUpload = false;
        }
        setTimeout(function () {
            var uploadBtn = document.querySelector('.e-file-select-wrap
button');
            uploadBtn.click();
        }, 100);
    }
}, '#file_tb_upload');
document.getElementById('file_tb_upload').onclick = function (args) {
    args.stopPropagation();
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 File Manager</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Essential JS 2 File Manager
Component">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
filemanager/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```

```
<div id="file"></div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Physical file service provider

To achieve the directory upload in the physical file service provider, use the below code snippet in `IActionResult Upload` method in the `Controllers/FileManagerController.cs` file.

```
`ts
[Route("Upload")]
public IActionResult Upload(string path, IList<IFormFile> uploadFiles, string action)
{
    FileManagerResponse uploadResponse;
    foreach (var file in uploadFiles)
    {
        var folders = (file.FileName).Split('/');
        // checking the folder upload
        if (folders.Length > 1)
        {
            for (var i = 0; i < folders.Length - 1; i++)
            {
                string newDirectoryPath = Path.Combine(this.basePath + path, folders[i]);
                if (!Directory.Exists(newDirectoryPath))
                {
                    this.operation.ToCamelCase(this.operation.Create(path, folders[i]));
                }
                path += folders[i] + "/";
            }
        }
        uploadResponse = operation.Upload(path, uploadFiles, action, null);
        if (uploadResponse.Error != null)
```

```

{
Response.Clear();
Response.ContentType = "application/json; charset=utf-8";
Response.StatusCode = Convert.ToInt32(uploadResponse.Error.Code);
Response.HttpContext.Features.Get<IHttpResponseFeature>().ReasonPhrase =
uploadResponse.Error.Message;
}
return Content("");
}
`

```

Refer to the [GitHub](#) for more details

And also add the below code snippet in `FileManagerResponse Upload` method in `Models/PhysicalFileProvider.cs` file.

```

`ts
string[] folders = name.Split('/');
string fileName = folders[folders.Length - 1];
var fullName = Path.Combine((this.contentRootPath + path), fileName);
`

```

Refer to the [GitHub](#) for more details.

Azure file service provider

For Azure file service provider, no customizations are needed for directory upload with server side and this will work with the below default upload method code.

Refer to the [GitHub](#) for more details.

NodeJS file service provider

To perform the directory upload in the NodeJS file service provider, use the below code snippet in `app.post` method in the `filesystem-server.js` file.

```

`ts
var folders = (req.body.filename).split('/');
var filepath = req.body.path;
var uploadedFileName = folders[folders.length - 1];
// checking the folder upload
if (folders.length > 1)
{
for (var i = 0; i < folders.length - 1; i++)
{

```

```

var newDirectoryPath = path.join(contentRootPath + filepath, folders[i]);
if (!fs.existsSync(newDirectoryPath)) {
  fs.mkdirSync(newDirectoryPath);
  (async () => {
    await FileManagerDirectoryContent(req, res, newDirectoryPath).then(data => {
      response = { files: data };
      response = JSON.stringify(response);
    });
  })();
}
filepath += folders[i] + "/";
}

fs.rename('./' + uploadedFileName, path.join(contentRootPath, filepath + uploadedFileName), function
(err) {
  if (err) {
    if (err.code !== 'EBUSY') {
      errorValue.message = err.message;
      errorValue.code = err.code;
    }
  }
});
}
`

```

Refer to the [GitHub](#) for more details.

Amazon file service provider

To perform the directory upload in the Amazon file service provider, use the below code snippet in `IActionResult AmazonS3Upload` method in the `Controllers/AmazonS3ProviderController.cs` file.

```

`ts
foreach (var file in uploadFiles)
{
  var folders = (file.FileName).Split('/');
  // checking the folder upload
  if (folders.Length > 1)
  {

```

```

for (var i = 0; i < folders.Length - 1; i++)
{
    if (!this.operation.checkFileExist(path, folders[i]))
    {
        this.operation.ToCamelCase(this.operation.Create(path, folders[i], dataObject));
    }
    path += folders[i] + "/";
}
}
}
,

```

Refer to the [GitHub](#) for more details.

And also add the below code snippet in `AsyncUpload` method in `Models/AmazonS3FileProvider.cs` file.

```

`ts
string[] folders = file.FileName.Split('/');
string name = folders[folders.Length - 1];
,

```

Refer to the [GitHub](#) for more details.

File operation request and response Parameters

The default parameters available in file operation request from the file manager and the corresponding response parameters required by the file manager are listed as follows.

Read

The following table represents the request parameters of *read* operations.

Parameter	Type	Default	Explanation
---- ---- ---- ----			
action	String	read	Name of the file operation.
path	String	-	Relative path from which the data has to be read.
showHiddenItems	Boolean	-	Defines show or hide the hidden items.
data	FileManagerDirectoryContent	-	Details about the current path (directory).

**Refer [File request and response contents](#) for the contents of data*.*

Example:

```

`ts
{

```



```
action: "read",
path: "/",
showHiddenItems: false,
data: []
},
,
```

The following table represents the response parameters of *read* operations.

Parameter	Type	Default	Explanation
----	----	----	----
cwd	FileManagerDirectoryContent	-	Path (Current Working Directory) details.
files	FileManagerDirectoryContent[]	-	Details of files and folders present in given path or directory.
error	ErrorDetails	-	Error Details

**Refer [File request and response contents](#) for the contents of cwd, files, and error*.*

Example:

```
`ts
{
  cwd:
  {
    name:"Download",
    size:0,
    dateModified:"2019-02-28T03:48:19.8319708+00:00",
    dateCreated:"2019-02-27T17:36:15.812193+00:00",
    hasChild:false,
    isFile:false,
    type:"",
    filterPath:"//Download//"
  },
  files:[
  {
    name:"Sample Work Sheet.xlsx",
    size:6172,
    dateModified:"2019-02-27T17:23:50.9651206+00:00",
    dateCreated:"2019-02-27T17:36:15.8151955+00:00",
```

```
hasChild:false,
isFile:true,
type:".xlsx",
filterPath:"//Download//"
}
],
error:null,
details:null
}
、
```

Create

The following table represents the request parameters of *create* operations.

Parameter	Type	Default	Explanation
----	----	----	----
action	String	create	Name of the file operation.
path	String	-	Relative path in which the folder has to be created.
name	String	-	Name of the folder to be created.
data	FileManagerDirectoryContent	-	Details about the current path (directory).

Refer [File request and response contents](#) for the contents of data

Example:

```
`ts
{
  action: "create",
  data: [
    {
      dateCreated: "2019-02-27T17:36:15.6571949+00:00",
      dateModified: "2019-03-12T10:17:31.8505975+00:00",
      filterPath: "/",
      hasChild: true,
      isFile: false,
      name: files,
      nodeId: "fe_tree",
      size: 0,
```

```
type: ""
}
],
name: "Hello",
path: "/"
}
`
```

The following table represents the response parameters of *create* operations.

Parameter	Type	Default	Explanation
files	FileManagerDirectoryContent[]	-	Details of the created folder
error	ErrorDetails	-	Error Details

**Refer [File request and response contents](#) for the contents of files and error*.*

Example:

```
`ts
{
  cwd: null,
  files: [
    {
      dateCreated: "2019-03-15T10:25:05.3596171+00:00",
      dateModified: "2019-03-15T10:25:05.3596171+00:00",
      filterPath: null,
      hasChild: false,
      isFile: false,
      name: "New",
      size: 0,
      type: ""
    }
  ],
  details: null,
  error: null
}
`
```

Rename

The following table represents the request parameters of *rename* operations.

Parameter	Type	Default	Explanation
----	----	----	----
action	String	rename	Name of the file operation.
path	String	-	Relative path in which the item is located.
name	String	-	Current name of the item to be renamed.
newname	String	-	New name for the item.
data	FileManagerDirectoryContent	-	Details of the item to be renamed.

**Refer [File request and response contents](#) for the contents of data*.*

Example:

```
`ts
{
  action: "rename",
  data: [
    {
      dateCreated: "2019-03-20T05:22:34.621Z",
      dateModified: "2019-03-20T08:45:56.000Z",
      filterPath: "/Pictures/Nature/",
      hasChild: false,
      iconClass: "e-fe-image",
      isFile: true,
      name: "seaviews.jpg",
      size: 95866,
      type: ".jpg"
    }
  ],
  newname: "seaview.jpg",
  name: "seaviews.jpg",
  path: "/Pictures/Nature/"
}
```

The following table represents the response parameters of *rename* operations.

Parameter	Type	Default	Explanation
files	FileManagerDirectoryContent[]	-	Details of the renamed item.
error	ErrorDetails	-	Error Details

**Refer [File request and response contents](#) for the contents of files and error*.*

Example:

```
`ts
{
  cwd:null,
  files:[
    {
      name:"seaview.jpg",
      size:95866,
      dateModified:"2019-03-20T08:45:56+00:00",
      dateCreated:"2019-03-20T05:22:34.6214847+00:00",
      hasChild:false,
      isFile:true,
      type:".jpg",
      filterPath:"//Pictures//Nature//seaview.jpg"
    }
  ],
  error:null,
  details:null
}
```

Delete

The following table represents the request parameters of *delete* operations.

Parameter	Type	Default	Explanation
action	String	delete	Name of the file operation.
path	String	-	Relative path where the items to be deleted are located.
names	String[]	-	List of the items to be deleted.
data	FileManagerDirectoryContent	-	Details of the item to be deleted.

**Refer [File request and response contents](#) for the contents of data*.*

Example:

```
`ts
{
  action: "delete",
  path: "/Hello/",
  names: ["New"],
  data: []
}
```

The following table represents the response parameters of *delete* operations.

Parameter	Type	Default	Explanation
files	FileManagerDirectoryContent[]	-	Details about the deleted item(s).
error	ErrorDetails	-	Error Details

**Refer [File request and response contents](#) for the contents of files and error*.*

Example:

```
`ts
{
  cwd: null,
  details: null,
  error: null,
  files: [
    {
      dateCreated: "2019-03-15T10:13:30.346309+00:00",
      dateModified: "2019-03-15T10:13:30.346309+00:00",
      filterPath: "/Hello/folder",
      hasChild: true,
      isFile: false,
      name: "folder",
      size: 0,
      type: ""
    }
  ]
}
```

```
]
}
`
```

Details

The following table represents the request parameters of *details* operations.

Parameter	Type	Default	Explanation
----	----	----	----
action	String	details	Name of the file operation.
path	String	-	Relative path where the items are located.
names	String[]	-	List of the items to get details.
data	FileManagerDirectoryContent	-	Details of the selected item.

**Refer [File request and response contents](#) for the contents of data*.*

Example:

```
`ts
{
  action: "details",
  path: "/FileContents/",
  names: ["All Files"],
  data: []
}
`
```

The following table represents the response parameters of *details* operations.

Parameter	Type	Default	Explanation
----	----	----	----
details	FileManagerDirectoryContent	-	Details of the requested item(s).
error	ErrorDetails	-	Error Details

**Refer [File request and response contents](#) for the contents of details and error*.*

Example:

```
`ts
{
  cwd:null,
  files:null,
  error:null,
}
```

details:

```
{
  name:"All Files",
  location:"//Files//FileContents//All Files",
  isFile:false,
  size:"679.8 KB",
  created:"3/8/2019 10:18:37 AM",
  modified:"3/8/2019 10:18:39 AM",
  multipleFiles:false
}
}
```

[Search](#)

The following table represents the request parameters of *search* operations.

Parameter	Type	Default	Explanation
----	----	----	----
action	String	search	Name of the file operation.
path	String	-	Relative path to the directory where the files should be searched.
showHiddenItems	Boolean	-	Defines show or hide the hidden items.
caseSensitive	Boolean	-	Defines search is case sensitive or not.
searchString	String	-	String to be searched in the directory.
data	FileManagerDirectoryContent	-	Details of the searched item.

Example:

```
`ts
{
  action: "search",
  path: "/",
  searchString: "nature",
  showHiddenItems: false,
  caseSensitive: false,
  data: []
}
```


The following table represents the response parameters of *search* operations.

Parameter	Type	Default	Explanation
----	----	----	----
cwd	FileManagerDirectoryContent	-	Path (Current Working Directory) details.
files	FileManagerDirectoryContent[]	-	Files and folders in the searched directory that matches the search input.
error	ErrorDetails	-	Error Details

Refer [File request and response contents](#) for the contents of cwd, files and error.

Example:

```
`ts
{
  cwd:
  {
    name:files,
    size:0,
    dateModified:"2019-03-15T10:07:00.8658158+00:00",
    dateCreated:"2019-02-27T17:36:15.6571949+00:00",
    hasChild:true,
    isFile:false,
    type:"",
    filterPath:"/"
  },
  files:[
    {
      name:"Nature",
      size:0,
      dateModified:"2019-03-08T10:18:42.9937708+00:00",
      dateCreated:"2019-03-08T10:18:42.5907729+00:00",
      hasChild:true,
      isFile:false,
      type:"",
      filterPath:"//FileContents//Nature"
    }
  ]
}
```

```
],  
error:null,  
details:null  
}  
`
```

Copy

The following table represents the request parameters of *copy* operations.

Parameter	Type	Default	Explanation
action	String	copy	Name of the file operation.
path	String	-	Relative path to the directory where the files should be copied.
names	String[]	-	List of files to be copied.
targetPath	String	-	Relative path where the items to be pasted are located.
data	FileManagerDirectoryContent	-	Details of the copied item.
renameFiles	String[]	-	Details of the renamed item.

Example:

```
`ts  
{  
  action: "copy",  
  path: "/",  
  names: ["6.png"],  
  renameFiles: ["6.png"],  
  targetPath: "/Videos/"  
}  
`
```

The following table represents the response parameters of *copy* operations.

Parameter	Type	Default	Explanation
cwd	FileManagerDirectoryContent	-	Path (Current Working Directory) details.
files	FileManagerDirectoryContent[]	-	Details of copied files or folders
error	ErrorDetails	-	Error Details

Refer [File request and response contents](#) for the contents of cwd, files and error.

Example:

```
`ts
{
  cwd:null,
  files:[
    {
      path:null,
      action:null,
      newName:null,
      names:null,
      name:"justin.mp4",
      size:0,
      previousName:"album.mp4",
      dateModified:"2019-06-21T06:58:32+00:00",
      dateCreated:"2019-06-24T04:22:14.6245618+00:00",
      hasChild:false,
      isFile:true,
      type:".mp4",
      id:null,
      filterPath:"/"
    }
  ],
  error:null,
  details:null
}
```

Move

The following table represents the request parameters of *move* operations.

Parameter	Type	Default	Explanation
-----	-----	-----	-----
action	String	move	Name of the file operation.
path	String	-	Relative path to the directory where the files should be copied.
names	String[]	-	List of files to be moved.
targetPath	String	-	Relative path where the items to be pasted are located.

|data|FileManagerDirectoryContent|-|Details of the moved item.|

|renameFiles|String[]|-|Details of the renamed item.|

Example:

```
`ts
{
  action: "move",
  path: "/",
  names: ["6.png"],
  renameFiles: ["6.png"],
  targetPath: "/Videos/"
}
```

The following table represents the response parameters of *copy* operations.

|Parameter|Type|Default|Explanation|

|----|----|----|----|

|cwd|FileManagerDirectoryContent|-|Path (Current Working Directory) details.|

|files|FileManagerDirectoryContent[]|-|Details of cut files or folders|

|error|ErrorDetails|-|Error Details|

Refer [File request and response contents](#) for the contents of cwd, files and error.

Example:

```
`ts
{
  cwd:null,
  files:[
    {
      path:null,
      action:null,
      newName:null,
      names:null,
      name:"justin biber.mp4",
      size:0,
      previousName:"justin biber.mp4",
      dateModified:"2019-06-21T06:58:32+00:00",
    }
  ]
}
```

```
dateCreated:"2019-06-24T04:26:49.2690476+00:00",
hasChild:false,
isFile:true,
type:".mp4",
id:null,
filterPath:"//Videos//"
},
error:null,
details:null
},
`
```

Upload

The following table represents the request parameters of *Upload* operations.

Parameter	Type	Default	Explanation
----	----	----	----
action	String	Save	Name of the file operation.
path	String	-	Relative path to the location where the file has to be uploaded.
uploadFiles	<code>IList<IFormFile></code>	-	File that are uploaded.

Example:

```
`ts
uploadFiles: (binary),
path: /,
action: Save,
data: {
path:null,
action:null,
newName:null,
names:null,
name:"Downloads",
size:0,
previousName:null,
dateModified:"2019-07-22T11:23:46.7153977 00:00",
```

```
dateCreated:"2019-07-22T11:26:13.9047229 00:00",
hasChild:false,
isFile:false,
type:"",
id:null,
filterPath:"/",
targetPath:null,
renameFiles:null,
uploadFiles:null,
caseSensitive:false,
searchString:null,
showHiddenItems:false,
fmiconClass:null,
fmid:"fetree1",
fmpld:null,
fmselected:false,
fmicon:null,
data:null,
targetData:null,
permission:null
}
`
```

The upload response is an empty string.

[Download](#)

The following table represents the request parameters of *download* operations.

Parameter	Type	Default	Explanation
action	String	download	Name of the file operation
path	String	-	Relative path to location where the files to download are present.
names	String[]	-	Name list of the items to be downloaded.
data	FileManagerDirectoryContent	-	Details of the download item.

Example:

```
`ts
```

```
{
  action:"download",
  path:"/",
  names:["1.png"],
  data:[
    {
      path:null,
      action:null,
      newName:null,
      names:null,
      name:"1.png",
      size:49792,
      previousName:null,
      dateModified:"2019-07-22T12:15:45.0972405+00:00",
      dateCreated:"2019-07-22T12:15:45.0816042+00:00",
      hasChild:false,
      isFile:true,
      type:".png",
      id:null,
      filterPath:"/",
      targetPath:null,
      renameFiles:null,
      uploadFiles:null,
      caseSensitive:false,
      searchString:null,
      showHiddenItems:false,
      fmiconClass:"e-fe-image",
      fmid:null,
      fmpId:null,
      fmselected:false,
      fmicon:null,
      data:null,
      targetData:null,
```

```

permission:null,
fmcreated:"2019-07-22T12:15:45.081Z",
fmmodified:"2019-07-22T12:15:45.097Z",
fmimageUrl:"https://ej2-aspcore-service.azurewebsites.net/api/FileManager/GetImage?path=/1.png",
fmimageAttr:
{
  alt:"1.png"
},
fmhtmlAttr:
{
  class:"e-large-icon",
  title:"1.png"
}
]
}
,

```

Downloads the requested items from the file server in response.

[GetImage](#)

The following table represents the request parameters of *GetImage* operations.

Parameter	Type	Default	Explanation
path	String	-	Relative path to the image file

Return the image as a file stream in response.

The request from the file manager can be customized using the `beforeSend` event. Additional information can be passed to the file manager in file operation response and can be used in customization.

[File request and response contents](#)

The following table represents the contents of *data*, *cwd*, and *files* in the file manager request and response.

Parameter	Type	Default	Explanation
name	String	-	File name
dateCreated	String	-	Date in which file was created (UTC Date string).
dateModified	String	-	Date in which file was last modified (UTC Date string).

filterPath	String	-	Relative path to the file or folder.
hasChild	Boolean	-	Defines this folder has any child folder or not.
isFile	Boolean	-	Say whether the item is file or folder.
size	Number	-	File size
type	String	-	File extension

The following table represents the contents of *error* in the file manager request and response.

Parameter	Type	Default	Explanation
code	String		Error code
message	String		Error message
fileExists	String[]		List of duplicate file names

The following table represents the contents of *details* in the file manager request and response.

Parameter	Type	Default	Explanation
name	String		File name
dateCreated	String		Date in which file was created (UTC Date string).
dateModified	String		Date in which file was last modified (UTC Date string).
filterPath	String		Relative path to the file or folder.
hasChild	Boolean		Defines this folder has any child folder or not.
isFile	Boolean		Say whether the item is file or folder.
size	Number		File size
type	String		File extension
multipleFiles	Boolean		Say whether the details are about single file or multiple files.

Action Buttons

The file manager has several menu buttons to access the file operations. The list of menu buttons available in the file manager is given in the following table.

Menu Button	Behaviour
SortBy	Opens the sub menu to choose the sorting order and sorting parameter.
View	Opens the sub menu to choose the View.
Open	Navigates to the selected folder. Opens the preview for image files.
Refresh	Initiates the read operation for the current directory and displays the updated directory content.
NewFolder	Opens the new folder dialog box to receive the name for the new folder.

|Rename| Opens the rename dialog box to receive the new name for the selected item.|

|Delete| Opens the delete dialog box to confirm the removal of the selected items from the file system.|

|Upload| Opens the upload box to select the items to upload to the file system.|

|Download| Downloads the selected item(s).|

|Details| Get details about the selected items and display them in details dialog box.|

|SelectAll| Selects all the files and folders displayed in the view section.|

The action menu buttons are present in the toolbar and context menu. The toolbar contains the buttons based on the selected items count, while the context menu will appear with a list based on the target.

Toolbar

The toolbar can be divided into two sections as right and left. Whenever the toolbar buttons exceed the size, the buttons present in the left section of the toolbar will be moved to the toolbar popup.

The following table provides the toolbar buttons that appear based on the selection.

<!-- markdownlint-disable MD033 -->

Selected Items Count	Left section	Right section
0 (none of the item)	<i>SortBy</i> Refresh <i>NewFolder</i> Upload	<i>View</i> Details
1 (single item selected)	<i>Delete</i> Download* Rename	<i>Selected items count</i> View* Details
>1 (multiple selection)	<i>Delete</i> Download	<i>Selected items count</i> View* Details

Context menu

The following table provides the default context menu item and the corresponding target areas.

<!-- markdownlint-disable MD033 -->

Menu Name	Menu Items	Target
Layout	<i>SortBy</i> View <i>Refresh</i> <i>NewFolder</i> Upload Details* Select all	<i>Empty space in the view section (details view and large icon view area)</i> . Empty folder content.
Folders	<i>Open</i> Delete <i>Rename</i> Downloads* Details	* Folders in treeview, details view, and large icon view.
Files	<i>Open</i> Delete <i>Rename</i> Downloads* Details	* Files in details view and large icon view.

Views in ##Platform_Name## File manager control

View is the section where the files and folders are displayed for the user to browse. The [view](#) API can also be used to change the initial view of the file manager.

The file manager has two types of [views](#) to display the files and folders.

- [Largelcons View](#)

- [Details View](#)

LargeIcons View

By Default, File Manager is rendered with largeicons view. The following example demonstrate this.

INDEX.JS

```
var hostUrl = 'https://ej2-aspcore-service.azurewebsites.net/';  
// inject feature modules of the file manager  
ej.filemanager.FileManager.Inject(ej.filemanager.Toolbar,ej.filemanager.NavigationPane);  
// initialize File Manager component  
var filemanagerInstance = new ej.filemanager.FileManager({  
  ajaxSettings: {  
    url: hostUrl + 'api/FileManager/FileOperations',  
    getImageUrl: hostUrl + 'api/FileManager/GetImage',  
    uploadUrl: hostUrl + 'api/FileManager/Upload',  
    downloadUrl: hostUrl + 'api/FileManager/Download'  
  },  
});  
// render initialized File Manager  
filemanagerInstance.appendTo('#filemanager');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>  
  <title>Essential JS 2 File Manager</title>  
  <meta charset="utf-8">  
  <meta name="viewport" content="width=device-width, initial-scale=1.0">  
  <meta name="description" content="Essential JS 2 File Manager  
Component">  
  <meta name="author" content="Syncfusion">  
  <link href="index.css" rel="stylesheet">  
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-  
base/styles/material.css" rel="stylesheet">  
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-  
inputs/styles/material.css" rel="stylesheet">  
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-  
popups/styles/material.css" rel="stylesheet">  
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-  
buttons/styles/material.css" rel="stylesheet">  
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-  
splitbuttons/styles/material.css" rel="stylesheet">  
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-  
layouts/styles/material.css" rel="stylesheet">  
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-  
navigations/styles/material.css" rel="stylesheet">  
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-  
grids/styles/material.css" rel="stylesheet">  
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-  
filemanager/styles/material.css" rel="stylesheet">  
  
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"  
type="text/javascript"></script>
```

```
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="filemanager"></div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Details View

Details view is an injectable module in the file manager so, it should be injected before rendering the file manager to avail its functionality. The default appearance of the file manager can be changed from largeicons to details view by using the [view](#) property. The following example demonstrate the file manager with details view.

INDEX.JS

```
var hostUrl = 'https://ej2-aspcore-service.azurewebsites.net/';
// inject feature modules of the file manager
ej.filemanager.FileManager.Inject(ej.filemanager.DetailsView,ej.filemanager.
Toolbar,ej.filemanager.NavigationPane);
// initialize File Manager component
var filemanagerInstance = new ej.filemanager.FileManager({
    ajaxSettings: {
        url: hostUrl + 'api/FileManager/FileOperations',
        getImageUrl: hostUrl + 'api/FileManager/GetImage',
        uploadUrl: hostUrl + 'api/FileManager/Upload',
        downloadUrl: hostUrl + 'api/FileManager/Download'
    },
    // Initial view of File Manager is set to details view
    view: "Details"
});
// render initialized File Manager
filemanagerInstance.appendTo('#filemanager');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 File Manager</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Essential JS 2 File Manager
Component">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
filemanager/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="filemanager"></div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customization in ##Platform_Name## File manager control

The file manager component allows customizing its functionalities like, context menu, searching, uploading, toolbar using APIs. Given below are some of the functionalities that can be customized in the File Manager,

- [Context menu customization](#)
- [Details view customization](#)
- [Navigation pane customization](#)
- [Show/Hide file extension](#)
- [Show/Hide hidden items](#)
- [Show/Hide thumbnail images in large icons view](#)
- [Toolbar customization](#)
- [Upload customization](#)
- [Tooltip customization](#)

Context menu customization

The context menu settings like, items to be displayed on files, folders and layout click and visibility can be customized using [contextMenuSettings](#) property.

INDEX.JS

```

var baseUrl = 'https://ej2-aspcore-service.azurewebsites.net/';
// inject feature modules of the file manager
ej.filemanager.FileManager.Inject(ej.filemanager.DetailsView,ej.filemanager.
Toolbar,ej.filemanager.NavigationPane);
// initialize File Manager component
var filemanagerInstance = new ej.filemanager.FileManager({
  ajaxSettings: {
    url: baseUrl + 'api/FileManager/FileOperations',
    getImageUrl: baseUrl + 'api/FileManager/GetImage',
    uploadUrl: baseUrl + 'api/FileManager/Upload',
    downloadUrl: baseUrl + 'api/FileManager/Download'
  },
  // Context Menu settings customization
  contextMenuSettings: { file: ['Open', '|', 'Details'], folder: ['Open',
'|', 'Details'], layout: ['SortBy', 'View', 'Refresh', '|', 'Details', '|'],
visible: true}
});
// render initialized File Manager
filemanagerInstance.appendTo('#filemanager');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 File Manager</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 File Manager
Component">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
filemanager/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>

```

```

</head>
<body>

    <div id="filemanager"></div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Details view customization

The details view settings like, column width, header text, template for each field can be customized using [detailsViewSettings](#) property.

INDEX.JS

```

var baseUrl = 'https://ej2-aspcore-service.azurewebsites.net/';
// inject feature modules of the file manager
ej.filemanager.FileManager.Inject(ej.filemanager.DetailsView,ej.filemanager.
Toolbar,ej.filemanager.NavigationPane);
// initialize File Manager component
var filemanagerInstance = new ej.filemanager.FileManager({
    ajaxSettings: {
        url: baseUrl + 'api/FileManager/FileOperations',
        getImageUrl: baseUrl + 'api/FileManager/GetImage',
        uploadUrl: baseUrl + 'api/FileManager/Upload',
        downloadUrl: baseUrl + 'api/FileManager/Download'
    },
    // Initial view of File Manager is set to details view
    view: "Details",
    // Details View settings customization
    detailsViewSettings: {
        columns: [
            {field: 'name', headerText: 'File Name', minWidth: 120, width:
'auto', customAttributes: { class: 'e-fe-grid-name' },template: '${name}'},
            {field: 'size', headerText: 'File Size',minWidth: 50, width:
'110', template: '${size}'},
            { field: '_fm_modified', headerText: 'Date Modified',minWidth:
50, width: '190'}
        ]
    }
});
// render initialized File Manager
filemanagerInstance.appendTo('#filemanager');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 File Manager</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

```

```

<meta name="description" content="Essential JS 2 File Manager
Component">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
filemanager/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="filemanager"></div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Navigation pane customization

The navigation pane settings like, minimum and maximum width and visibility can be customized using [navigationPaneSettings](#) property.

INDEX.JS

```

var hostUrl = 'https://ej2-aspcore-service.azurewebsites.net/';
// inject feature modules of the file manager
ej.filemanager.FileManager.Inject(ej.filemanager.DetailsView,ej.filemanager.
Toolbar,ej.filemanager.NavigationPane);
// initialize File Manager component
var filemanagerInstance = new ej.filemanager.FileManager({
    ajaxSettings: {
        url: hostUrl + 'api/FileManager/FileOperations',
        getImageUrl: hostUrl + 'api/FileManager/GetImage',

```



```

        uploadUrl: hostUrl + 'api/FileManager/Upload',
        downloadUrl: hostUrl + 'api/FileManager/Download'
    },
    // Navigation Pane settings customization
    navigationPaneSettings: { maxWidth: '850px', minWidth: '140px', visible:
true}
});
// render initialized File Manager
filemanagerInstance.appendTo('#filemanager');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 File Manager</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Essential JS 2 File Manager
Component">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
filemanager/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="filemanager"></div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Show/Hide file extension

The file extensions are displayed in the File Manager by default. This can be hidden by disabling the [showFileExtension](#) property.

In File Manager [fileLoad](#) and [fileOpen](#) events are triggered before the file/folder is rendered and before the file/folder is opened respectively. These events can be utilized to perform operations before a file/folder is rendered or opened.

INDEX.JS

```
var hostUrl = 'https://ej2-aspcore-service.azurewebsites.net/';
// inject feature modules of the file manager
ej.filemanager.FileManager.Inject(ej.filemanager.DetailsView,ej.filemanager.
Toolbar,ej.filemanager.NavigationPane);
// initialize File Manager component
var filemanagerInstance = new ej.filemanager.FileManager({
  ajaxSettings: {
    url: hostUrl + 'api/FileManager/FileOperations',
    getImageUrl: hostUrl + 'api/FileManager/GetImage',
    uploadUrl: hostUrl + 'api/FileManager/Upload',
    downloadUrl: hostUrl + 'api/FileManager/Download'
  },
  // Hides the file extension in File Manager
  showFileExtension: false,
  // File Manager's beforeFileLoad event
  fileLoad: onBeforeFileLoad,
  // File Manager's beforeFileOpen event
  fileOpen: onBeforeFileOpen
});
// render initialized FileManager
filemanagerInstance.appendTo('#filemanager');
// File Manager's file beforeFileLoad function
function onBeforeFileLoad(args) {
  console.log(args.fileDetails.name + " is loading");
}
// File Manager's file beforeFileOpen function
function onBeforeFileOpen(args) {
  console.log(args.fileDetails.name + " is opened");
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 File Manager</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 File Manager
Component">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
filemanager/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="filemanager"></div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Show/Hide hidden items

The File Manager provides support to show/hide the hidden items by enabling/disabling the [showHiddenItems](#) property.

INDEX.JS

```

var hostUrl = 'https://ej2-aspcore-service.azurewebsites.net/';
// inject feature modules of the file manager
ej.filemanager.FileManager.Inject(ej.filemanager.DetailsView,ej.filemanager.
Toolbar,ej.filemanager.NavigationPane);
// initialize File Manager component
var filemanagerInstance = new ej.filemanager.FileManager({
    ajaxSettings: {
        url: hostUrl + 'api/FileManager/FileOperations',
        getImageUrl: hostUrl + 'api/FileManager/GetImage',
        uploadUrl: hostUrl + 'api/FileManager/Upload',
        downloadUrl: hostUrl + 'api/FileManager/Download'
    },
    // The default value set for showHiddenItems is false
    showHiddenItems: true
});
// render initialized File Manager
filemanagerInstance.appendTo('#filemanager');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 File Manager</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 File Manager
Component">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
filemanager/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="filemanager"></div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Show/Hide thumbnail images in large icons view

The thumbnail images are displayed in the File Manager's large icons view by default. This can be hidden by disabling the [showThumbnail](#) property.

INDEX.JS

```

var hostUrl = 'https://ej2-aspcore-service.azurewebsites.net/';

```

```
// inject feature modules of the file manager
ej.filemanager.FileManager.Inject(ej.filemanager.DetailsView,ej.filemanager.
Toolbar,ej.filemanager.NavigationPane);
// initialize File Manager component
var filemanagerInstance = new ej.filemanager.FileManager({
  ajaxSettings: {
    url: hostUrl + 'api/FileManager/FileOperations',
    getImageUrl: hostUrl + 'api/FileManager/GetImage',
    uploadUrl: hostUrl + 'api/FileManager/Upload',
    downloadUrl: hostUrl + 'api/FileManager/Download'
  },
  // Hides the thumbnail images in File Manager's large icons view
  showThumbnail: false
});
// render initialized File Manager
filemanagerInstance.appendTo('#filemanager');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 File Manager</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 File Manager
Component">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
filemanager/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="filemanager"></div>
</script>
```

```

var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Toolbar customization

The toolbar settings like, items to be displayed in toolbar and visibility can be customized using [toolbarSettings](#) property.

INDEX.JS

```

var hostUrl = 'https://ej2-aspcore-service.azurewebsites.net/';
// inject feature modules of the file manager
ej.filemanager.FileManager.Inject(ej.filemanager.DetailsView,ej.filemanager.
Toolbar,ej.filemanager.NavigationPane);
// initialize File Manager component
var filemanagerInstance = new ej.filemanager.FileManager({
    ajaxSettings: {
        url: hostUrl + 'api/FileManager/FileOperations',
        getImageUrl: hostUrl + 'api/FileManager/GetImage',
        uploadUrl: hostUrl + 'api/FileManager/Upload',
        downloadUrl: hostUrl + 'api/FileManager/Download'
    },
    // Toolbar settings customization
    toolbarSettings: { items: ['NewFolder', 'Refresh', 'View', 'Details'],
visible: true}
});
// render initialized File Manager
filemanagerInstance.appendTo('#filemanager');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 File Manager</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Essential JS 2 File Manager
Component">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
filemanager/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="filemanager"></div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [How to add new items or customize default items](#)

Upload customization

The upload settings like, minimum and maximum file size and enabling auto upload can be customized using [uploadSettings](#) property.

INDEX.JS

```

var hostUrl = 'https://ej2-aspcore-service.azurewebsites.net/';
// inject feature modules of the file manager
ej.filemanager.FileManager.Inject(ej.filemanager.DetailsView,ej.filemanager.
Toolbar,ej.filemanager.NavigationPane);
// initialize File Manager component
var filemanagerInstance = new ej.filemanager.FileManager({
    ajaxSettings: {
        url: hostUrl + 'api/FileManager/FileOperations',
        getImageUrl: hostUrl + 'api/FileManager/GetImage',
        uploadUrl: hostUrl + 'api/FileManager/Upload',
        downloadUrl: hostUrl + 'api/FileManager/Download'
    },
    // Upload settings customization
    uploadSettings: { maxFileSize: 233332, minFileSize: 120, autoUpload:
true}
});
// render initialized File Manager
filemanagerInstance.appendTo('#filemanager');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 File Manager</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 File Manager
Component">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
filemanager/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="filemanager"></div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Tooltip customization

The tooltip value can be customized by adding extra content to the title of the toolbar, navigation pane, details view and large icons of the file manager element.

INDEX.JS

```

var hostUrl = 'https://ej2-aspcore-service.azurewebsites.net/';
// inject feature modules of the file manager

```



```

ej.filemanager.FileManager.Inject(ej.filemanager.DetailsView,ej.filemanager.
Toolbar,ej.filemanager.NavigationPane);
// initialize File Manager component
var fileObj = new ej.filemanager.FileManager({
  ajaxSettings: {
    url: hostUrl + 'api/FileManager/FileOperations',
    uploadUrl: hostUrl + 'api/FileManager/Upload',
    downloadUrl: hostUrl + 'api/FileManager/Download',
    getImageUrl: hostUrl + 'api/FileManager/GetImage'
  },
  created: function () { addTooltip(); },
  fileLoad: function (args) {
    //Native tooltip customization to display additonal information in new
line
    var target = args.element;
    if (args.module === 'DetailsView') {
      var ele = select('[title]', args.element);
      var title = args.fileDetails.name +
        '\n' + args.fileDetails.dateModified;
      ele.setAttribute('title', title);
    }
    else if (args.module === 'LargeIconsView') {
      var title = args.fileDetails.name +
        '\n' + args.fileDetails.dateModified;
      target.setAttribute('title', title);
    }
  }
});
// render initialized File Manager
fileObj.appendTo('#filemanager');
function addTooltip() {
  var tooltip = new ej.popups.Tooltip({
    target: '#' + fileObj.element.id + '_toolbar [title]',
    beforeRender: onTooltipBeforeRender
  });
  tooltip.appendTo('#' + fileObj.element.id + '_toolbar');
}
function onTooltipBeforeRender(args) {
  var buttonId = args.target.children[0].id;
  var description = '';
  switch (buttonId) {
    case fileObj.element.id + '_tb_newfolder':
      description = 'Create a new folder';
      break;
    case fileObj.element.id + '_tb_upload':
      description = 'Upload new files';
      break;
    case fileObj.element.id + '_tb_cut':
      description = 'Cut files and folders from the current location';
      break;
    case fileObj.element.id + '_tb_copy':
      description = 'Copy files and folders from the current
location';
      break;
    case fileObj.element.id + '_tb_paste':
      description = 'Paste files and folders in the current location';
      break;
  }
}

```

```

        case fileObj.element.id + '_tb_delete':
            description = 'Delete selected files and folders';
            break;
        case fileObj.element.id + '_tb_download':
            description = 'Download selected files and folders';
            break;
        case fileObj.element.id + '_tb_rename':
            description = 'Rename selected file or folder';
            break;
        case fileObj.element.id + '_tb_sortby':
            description = 'Change the file sorting order';
            break;
        case fileObj.element.id + '_tb_refresh':
            description = 'Refresh the current location';
            break;
        case fileObj.element.id + '_tb_selection':
            description = 'Following items are currently selected: ';
            for (var i = 0; i < fileObj.selectedItems.length; i++) {
                description = description + '</br>' +
fileObj.selectedItems[i];
            }
            break;
        case fileObj.element.id + '_tb_view':
            description = 'Switch the layout view';
            break;
        case fileObj.element.id + '_tb_details':
            description = 'Get the details of the seletced items';
            break;
        default:
            description = '';
            break;
    }
    this.content = args.target.getAttribute('title') + '</br>' +
description;
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 File Manager</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Essential JS 2 File Manager
Component">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
filemanager/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="filemanager"></div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Multiple selection in ##Platform_Name## File manager control

The file manager allows you to select multiple files by enabling the [allowMultiSelection](#) property (enabled by default). The multiple selection can be done by pressing the **Ctrl** key or **Shift** key and selecting the files. The check box can also be used to do multiple selection. **Ctrl + A** can be used to select all files in the current directory. The [fileSelect](#) event will be triggered when the items of file manager control is selected or unselected.

INDEX.JS

```

var hostUrl = 'https://ej2-aspcore-service.azurewebsites.net/';
// inject feature modules of the file manager
ej.filemanager.FileManager.Inject(ej.filemanager.DetailsView,ej.filemanager.
Toolbar,ej.filemanager.NavigationPane);
// initialize File Manager component
var filemanagerInstance = new ej.filemanager.FileManager({
    ajaxSettings: {
        url: hostUrl + 'api/FileManager/FileOperations',
        getImageUrl: hostUrl + 'api/FileManager/GetImage',
        uploadUrl: hostUrl + 'api/FileManager/Upload',
        downloadUrl: hostUrl + 'api/FileManager/Download'
    },
    allowMultiSelection:true, // allowMultiSelection is true by default.
    // File Manager's file select event
    fileSelect: onFileSelect
});
// render initialized File Manager
filemanagerInstance.appendTo('#filemanager');
// File Manager's file select event function

```

```
function onFileSelect(args) {  
    console.log(args.fileDetails.name + " has been " + args.action + "ed");  
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>  
    <title>Essential JS 2 File Manager</title>  
    <meta charset="utf-8">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <meta name="description" content="Essential JS 2 File Manager  
Component">  
    <meta name="author" content="Syncfusion">  
    <link href="index.css" rel="stylesheet">  
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-  
base/styles/material.css" rel="stylesheet">  
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-  
inputs/styles/material.css" rel="stylesheet">  
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-  
popups/styles/material.css" rel="stylesheet">  
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-  
buttons/styles/material.css" rel="stylesheet">  
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-  
splitbuttons/styles/material.css" rel="stylesheet">  
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-  
layouts/styles/material.css" rel="stylesheet">  
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-  
navigations/styles/material.css" rel="stylesheet">  
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-  
grids/styles/material.css" rel="stylesheet">  
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-  
filemanager/styles/material.css" rel="stylesheet">  
  
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"  
type="text/javascript"></script>  
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type  
="text/javascript"></script>  
</head>  
<body>  
  
    <div id="filemanager"></div>  
<script>  
var ele = document.getElementById('container');  
if(ele) {  
    ele.style.visibility = "visible";  
}  
</script>  
<script src="index.js" type="text/javascript"></script>  
</body></html>
```

Note: The File Manager has support to select files and folders initially or dynamically by specifying their names in [selectedItems](#) property.

Drag and drop in ##Platform_Name## File manager control

The file manager allows files or folders to be moved from one folder to another by using the [allowDragAndDrop](#) property. It also supports uploading a file by dragging it from Windows Explorer to FileManager control. You can enable or disable this support by using the [allowDragAndDrop](#) property of file manager.

The event triggered in drag and drop support are

- [fileDragStart](#) - Triggers when the file/folder dragging is started.
- [fileDragging](#) - Triggers while dragging the file/folder.
- [fileDragStop](#) - Triggers when the file/folder is about to be dropped at the target.
- [fileDropped](#) - Triggers when the file/folder is dropped.

INDEX.JS

```
var hostUrl = 'https://ej2-aspcore-service.azurewebsites.net/';
// inject feature modules of the file manager
ej.filemanager.FileManager.Inject(ej.filemanager.DetailsView,ej.filemanager.
Toolbar,ej.filemanager.NavigationPane);
// initialize File Manager component
var filemanagerInstance = new ej.filemanager.FileManager({
  ajaxSettings: {
    url: hostUrl + 'api/FileManager/FileOperations',
    getImageUrl: hostUrl + 'api/FileManager/GetImage',
    uploadUrl: hostUrl + 'api/FileManager/Upload',
    downloadUrl: hostUrl + 'api/FileManager/Download'
  },
  allowDragAndDrop:true, // allowDragAndDrop is true by default.
  // File Manager's file drag start event
  fileDragStart: onFileDragStart,
  // File Manager's file dragging event
  fileDragging: onFileDragging,
  // File Manager's file drag stop event
  fileDragStop: onFileDragStop,
  // File Manager's file drag stop event
  fileDropped: onFileDropped
});
// render initialized File Manager
filemanagerInstance.appendTo('#filemanager');
// File Manager's file drag start event function
function onFileDragStart(args) {
  console.log("File drag start");
}
// File Manager's file drag stop event function
function onFileDragStop(args) {
  console.log("File drag stop");
}
// File Manager's file dragging event function
function onFileDragging(args) {
  console.log("File dragging");
}
// File Manager's file dropped event function
function onFileDropped(args) {
  console.log("File dropped");
}
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 File Manager</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 File Manager
Component">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
filemanager/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="filemanager"></div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

File system provider in ##Platform_Name## File manager control

The file system provider allows the File Manager component to manage the files and folders in a physical or cloud-based file system. It provides the methods for performing various file actions like creating a new folder, copying and moving of files or folders, deleting, uploading, and downloading the files or folders in the file system.

The following file providers are added in Syncfusion EJ2 File Manager component.

- [ASP.NET Core file system provider](#)
- [ASP.NET MVC 5 file system provider](#)
- [ASP.NET Core Azure cloud file system Provider](#)
- [ASP.NET MVC 5 Azure cloud file system Provider](#)
- [ASP.NET Core Amazon S3 cloud file provider](#)
- [ASP.NET MVC Amazon S3 cloud file provider](#)
- [File Transfer Protocol file system provider](#)
- [SQL database file system provider](#)
- [NodeJS file system provider](#)
- [Google Drive file system provider](#)
- [Firebase Realtime Database file system provider](#)
- [IBM Cloud Object Storage provider](#)

ASP.NET Core file system provider

The ASP.NET Core file system provider allows the users to access and manage the physical file system. To get started, clone the [ej2-aspcore-file-provider](#) using the following command.

```
`ts
git clone https://github.com/SyncfusionExamples/ej2-aspcore-file-provider ej2-aspcore-file-provider
cd ej2-aspcore-file-provider
`
```

After cloning, just open the project in Visual Studio and restore the NuGet packages. Now, set the root directory of the physical file system in the FileManager controller.

After setting the root directory of the file system, just build and run the project. Now, the project will be hosted in `http://localhost:{port}` and just mapping the **ajaxSettings** property of the FileManager component to the appropriate controller methods allows to manage the files in the physical file system.

```
`ts
let hostUrl = 'http://localhost:{port}/';
// Initializing File Manager ASP.NET Core service.
let filemanagerInstance: FileManager = new FileManager({
  ajaxSettings: {
    // Replace the hosted port number in the place of "{port}"
    url: hostUrl + "api/FileManager/FileOperations",
    downloadUrl: hostUrl + "api/FileManager/Download",
    uploadUrl: hostUrl + "api/FileManager/Upload",
    getImageUrl: hostUrl + "api/FileManager/GetImage"
  }
});
filemanagerInstance.appendTo('#filemanager');
```

Note: To learn more about the file actions that can be performed with ASP.NET Core file system provider, refer to this [link](#)

ASP.NET MVC 5 file system provider

The ASP.NET MVC 5 file system provider allows the users to access and manage the physical file system. To get started, clone the [ej2-aspmvc-file-provider](#) using the following command.

```
`ts
git clone https://github.com/SyncfusionExamples/ej2-aspmvc-file-provider ej2-aspmvc-file-provider
cd ej2-aspmvc-file-provider
`
```

After cloning, just open the project in Visual Studio and restore the NuGet packages. Now, set the root directory of the physical file system in the FileManager controller using the Root Folder method.

After setting the root directory of the file system, just build and run the project. Now, the project will be hosted in `http://localhost:{port}` and just mapping the **ajaxSettings** property of the FileManager component to the appropriate controller methods allows to manage the files in the physical file system.

```
`ts
let hostUrl = 'http://localhost:{port}/';
// Initializing File Manager ASP.NET MVC service.
let filemanagerInstance: FileManager = new FileManager({
  ajaxSettings: {
    // Replace the hosted port number in the place of "{port}"
    url: hostUrl + "FileManager/FileOperations",
    downloadUrl: hostUrl + "FileManager/Download",
    uploadUrl: hostUrl + "FileManager/Upload",
    getImageUrl: hostUrl + "FileManager/GetImage"
  }
});
filemanagerInstance.appendTo('#filemanager');
`
```

Note: To learn more about the file actions that can be performed with ASP.NET MVC 5 file system provider, refer to this [link](#)

ASP.NET Core Azure cloud file system provider

In ASP.NET Core, Azure file system provider allows the users to access and manage the blobs in the Azure blob storage. To get started, clone the [azure-aspcore-file-provider](#) using the following command

```
`ts
git clone https://github.com/SyncfusionExamples/azure-aspcore-file-provider azure-aspcore-file-provider
```


After cloning, just open the project in Visual Studio and restore the NuGet packages. Now, register the Azure storage by passing details like name, password, and blob name to the Register Azure method in the FileManager controller.

```
`ts
void RegisterAzure(string accountName, string accountKey, string blobName)
```

Then, set the blob container and the root blob directory by passing the corresponding URLs as parameters in the **setBlobContainer** method as follows.

```
`ts
void setBlobContainer(string blobPath, string filePath)
```

Note: Also, assign the same *blobPath URL* and *filePath URL* in [AzureFileOperations](#) and [AzureUpload](#) methods in the FileManager controller to determine the original path of the Azure blob.

After setting the blob container references, just build and run the project. Now, the project will be hosted in `http://localhost:{port}` and just mapping the **ajaxSettings** property of the FileManager component to the appropriate controller methods allows to manage the Azure blob storage.

```
`ts
let hostUrl = 'http://localhost:{port}/';
// Initializing File Manager Azure cloud file system service.
let filemanagerInstance: FileManager = new FileManager({
  ajaxSettings: {
    // Replace the hosted port number in the place of "{port}"
    url: hostUrl + "api/AzureProvider/AzureFileOperations",
    downloadUrl: hostUrl + "api/AzureProvider/AzureDownload",
    uploadUrl: hostUrl + "api/AzureProvider/AzureUpload",
    getImageUrl: hostUrl + "api/AzureProvider/AzureGetImage"
  }
});
filemanagerInstance.appendTo('#filemanager');
```

NuGet: Additionally, we have created a [NuGet](#) package of **ASP.NET Core Azure file system provider**.

Please, use the following command to install the NuGet package in an application.

```
`ts
dotnet add package Syncfusion.EJ2.FileManager.AzureFileProvider.AspNet.Core
```

Note: To learn more about the file actions that can be performed with ASP.NET Core Azure Cloud File System Provider, refer to this [link](#)

ASP.NET MVC Azure cloud file system provider

In ASP.NET MVC, Azure file system provider allows the users to access and manage the blobs in the Azure blob storage. To get started, clone the [ej2-azure-aspmvc-file-provider](#) using the following command

```
`ts
git clone https://github.com/SyncfusionExamples/ej2-azure-aspmvc-file-provider ej2-azure-aspmvc-file-provider
```

After cloning, just open the project in Visual Studio and restore the NuGet packages. Now, register the Azure storage by passing details like name, password, and blob name to the Register Azure method in the FileManager controller.

```
`ts
void RegisterAzure(string accountName, string accountKey, string blobName)
```

Then, set the blob container and the root blob directory by passing the corresponding URLs as parameters in the **setBlobContainer** method as follows.

```
`ts
void setBlobContainer(string blobPath, string filePath)
```

Note: Also, assign the same *blobPath URL* and *filePath URL* in [AzureFileOperations](#) and [AzureUpload](#) methods in the FileManager controller to determine the original path of the Azure blob.

After setting the blob container references, just build and run the project. Now, the project will be hosted in `http://localhost:{port}` and just mapping the **ajaxSettings** property of the FileManager component to the appropriate controller methods allows to manage the Azure blob storage.

```
`ts
let hostUrl = 'http://localhost:{port}';
// Initializing File Manager Azure cloud file system service.
let filemanagerInstance: FileManager = new FileManager({
  ajaxSettings: {
    // Replace the hosted port number in the place of "{port}"
    url: hostUrl + "AzureProvider/AzureFileOperations",
    downloadUrl: hostUrl + "AzureProvider/AzureDownload",
    uploadUrl: hostUrl + "AzureProvider/AzureUpload",
```

```

getImageUrl: hostUrl + "AzureProvider/AzureGetImage"
}
});
filemanagerInstance.appendTo('#filemanager');
`

```

Note: To learn more about the file actions that can be performed with ASP.NET MVC Azure Cloud File System Provider, refer to this [link](#)

ASP.NET Core Amazon S3 cloud file provider

In ASP.NET Core, Amazon **S3** (*Simple Storage Service*) cloud file provider allows the users to access and manage a server hosted file system as collection of objects stored in the Amazon S3 Bucket. To get started, clone the [amazon-s3-aspcore-file-provider](#) using the following command

```

`ts
git clone https://github.com/SyncfusionExamples/amazon-s3-aspcore-file-provider.git amazon-s3-aspcore-file-provider.git
`

```

Note: To learn more about creating and configuring an Amazon S3 bucket, refer to this [link](#).

After cloning, open the project in Visual Studio and restore the NuGet packages. Now, register Amazon S3 client account details like *awsAccessKeyId*, *awsSecretKeyId* and *awsRegion* details in

RegisterAmazonS3 method in the FileManager controller to perform the file operations.

```

`ts
void RegisterAmazonS3(string bucketName, string awsAccessKeyId, string awsSecretAccessKey, string bucketRegion)
`

```

After registering the Amazon client account details, just build and run the project. Now, the project will be hosted in `http://localhost:{port}` and just mapping the **ajaxSettings** property of the FileManager component to the appropriate controller methods allows to manage the Amazon **S3** (*Simple Storage Service*) bucket's objects storage.

```

`ts
let hostUrl = 'http://localhost:{port}/';
// Initializing File Manager with Amazon S3 service configuration.
let filemanagerInstance: FileManager = new FileManager({
// Replace the hosted port number in the place of "{port}"
ajaxSettings: {
url: hostUrl + "api/AmazonS3Provider/AmazonS3FileOperations",
downloadUrl: hostUrl + "api/AmazonS3Provider/AmazonS3Download",
uploadUrl: hostUrl + "api/AmazonS3Provider/AmazonS3Upload",

```

```
getImageUrl: hostUrl + "api/AmazonS3Provider/AmazonS3GetImage"
}
});
filemanagerInstance.appendTo('#filemanager');
`ts
```

Note: To learn more about the file actions that can be performed with Amazon S3 Cloud File provider, refer to this [link](#)

ASP.NET MVC Amazon S3 cloud file provider

In ASP.NET MVC, Amazon **S3** (*Simple Storage Service*) cloud file provider allows the users to access and manage a server hosted files as collection of objects stored in the Amazon S3 Bucket. To get started, clone the [ej2-amazon-s3-aspmvc-file-provider](#) using the following command

```
`ts
git clone https://github.com/SyncfusionExamples/ej2-amazon-s3-aspmvc-file-provider.git ej2-amazon-
s3-aspmvc-file-provider.git
`
```

Note: To learn more about creating and configuring an Amazon S3 bucket, refer to this [link](#).

After cloning, open the project in Visual Studio and restore the NuGet packages. Now, register Amazon S3 client account details like *awsAccessKeyId*, *awsSecretKeyId* and *awsRegion* details in

RegisterAmazonS3 method in the FileManager controller to perform the file operations.

```
`ts
void RegisterAmazonS3(string bucketName, string awsAccessKeyId, string awsSecretAccessKey, string
bucketRegion)
`
```

After registering the Amazon client account details, just build and run the project. Now, the project will be hosted in `http://localhost:{port}` and just mapping the **ajaxSettings** property of the FileManager component to the appropriate controller methods allows to manage the Amazon **S3** (*Simple Storage Service*) bucket's objects storage.

```
`ts
let hostUrl = 'http://localhost:{port}/';
// Initializing File Manager with Amazon S3 service configuration.
let filemanagerInstance: FileManager = new FileManager({
// Replace the hosted port number in the place of "{port}"
ajaxSettings: {
url: hostUrl + "FileManager/FileOperations",
downloadUrl: hostUrl + "FileManager/Download",
uploadUrl: hostUrl + "FileManager/Upload",
```

```

getImageUrl: hostUrl + "FileManager/GetImage"
}
});
filemanagerInstance.appendTo('#filemanager');
`

```

Note: To learn more about the file actions that can be performed with ASP.NET MVC Amazon S3 Cloud File Provider, refer to this [link](#)

File Transfer Protocol file system provider

In ASP.NET Core, File Transfer Protocol file system provider allows the users to access to the hosted file system as collection of objects stored in the file storage using File Transfer Protocol. To get started, clone the [ftp-aspcore-file-provider](#) using the following command

```

`ts
git clone https://github.com/SyncfusionExamples/ftp-aspcore-file-provider.git ftp-aspcore-file-
provider.git
`

```

After cloning, open the project in Visual Studio and restore the NuGet packages. Now, register File Transfer Protocol details like *hostName*, *userName* and *password* in **SetFTPConnection** method in the FileManager controller to perform the file operations.

```

`ts
void SetFTPConnection(string hostName, string userName, string password)
`

```

After registering the File Transfer Protocol details, just build and run the project. Now, the project will be hosted in `http://localhost:{port}` and just mapping the **ajaxSettings** property of the FileManager component to the appropriate controller methods allows to manage the FTP's objects storage.

```

`ts
let hostUrl = 'http://localhost:{port}/';
// Initializing File Manager with file transfer protocol service configuration.
let filemanagerInstance: FileManager = new FileManager({
// Replace the hosted port number in the place of "{port}"
ajaxSettings: {
url: hostUrl + "api/FTPProvider/FTPFileOperations",
downloadUrl: hostUrl + "api/FTPProvider/FTPDownload",
uploadUrl: hostUrl + "api/FTPProvider/FTPUpload",
getImageUrl: hostUrl + "api/FTPProvider/FTPGetImage"
}
});

```

```
filemanagerInstance.appendTo('#filemanager');
```

Note: To learn more about the file actions that can be performed with File Transfer Protocol file system provider, refer to this [link](#)

SQL database file system provider

In ASP.NET Core, SQL database file system provider in ASP.NET Core allows the users to manage the file system being maintained in a SQL database table. Unlike the other file system providers, the SQL database file system provider works on ID basis. Here, each file and folder have a unique ID based on which all the file operations will be performed. To get started, clone the [sql-server-database-aspcore-file-provider](#) using the following command.

```
`ts
```

```
<add name="FileExplorerConnection" connectionString="Data
Source=(LocalDB)\v11.0;AttachDbFilename=|DataDirectory|\FileManager.mdf;Integrated
Security=True;Trusted_Connection=true" />
```

After cloning, just open the project in Visual Studio and restore the NuGet packages. To establish the SQL server connection with the database file (for eg: FileManager.mdf), specify the connection string in the web config file as follows.

```
`ts
```

```
<add name="FileExplorerConnection" connectionString="Data
Source=(LocalDB)\v11.0;AttachDbFilename=|DataDirectory|\FileManager.mdf;Integrated
Security=True;Trusted_Connection=true" />
```

Then, make an entry for the connection string in `appsettings.json` file as follows.

```
`ts
```

```
"ConnectionStrings": {
  "FileManagerConnection": "Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=|DataDirectory|\\App_Data\\FileManager.mdf;In
tegrated Security=True;Connect Timeout=30"
}
```

Now, to configure the database connection, set the connection name, table name and root folder ID value by passing these values to the SetSQLConnection method.

```
`ts
```

```
void SetSQLConnection(string name, string tableName, string tableID)
```

Refer to this [FileManager.mdf](#), to learn about the pre-defined file system SQL database for the EJ2 File Manager.

After configuring the connection, just build and run the project. Now, the project will be hosted in `http://localhost:{port}` and just mapping the **ajaxSettings** property of the FileManager component to the appropriate controller methods allows to manage the files in the SQL database table.

```
`ts
```

```
let hostUrl = 'http://localhost:{port}/';  
// Initializing File Manager with SQL Server Database service configuration.  
let filemanagerInstance: FileManager = new FileManager({  
  ajaxSettings: {  
    // Replace the hosted port number in the place of "{port}"  
    url: hostUrl + "api/SQLProvider/SQLFileOperations",  
    downloadUrl: hostUrl + "api/SQLProvider/SQLDownload",  
    uploadUrl: hostUrl + "api/SQLProvider/SQLUpload",  
    getImageUrl: hostUrl + "api/SQLProvider/SQLGetImage"  
  }  
});  
filemanagerInstance.appendTo('#filemanager');
```

Note: To learn more about the file actions that can be performed with SQL database file system provider, refer to this [link](#)

NodeJS file system provider

The NodeJS file system provider allows the users to manage the files and folders in a physical file system. It provides methods for performing all basic file operations like creating a folder, copy, move, delete, and download files and folders in the file system. We can use of the NodeJS file system provider either by installing the [ej2-filemanager-node-filesystem](#) package or by cloning the [file system provider](#) from the GitHub.

Using ej2-filemanager-node-filesystem package

- Install the ej2-filemanager-node-filesystem package by running the below command.

```
`ts
```

```
npm install @syncfusion/ej2-filemanager-node-filesystem
```

- After installing the package, navigate to the ej2-filemanager-node-filesystem package folder within the node-modules.
- Run the command **npm install** command.

Cloning the ej2-filemanager-node-filesystem from GitHub

- Clone the ej2-filemanager-node-filesystem using the following command.

```
`ts
```

```
git clone https://github.com/SyncfusionExamples/ej2-filemanager-node-filesystem.git node-filesystem-provider
```

```
`
```

- After cloning, open the root folder and run the command **npm install** command.

After installing the packages, set the root folder directory of the physical file system in the package JSON under scripts sections as follows.

```
`ts
```

```
"start": "node filesystem-server.js -d D:/Projects"
```

```
`
```

Note: By default, the root directory will be configured to set **C:/Users** as the root directory.

To set the port in which the project to be hosted and the root directory of the file system. Run the following command.

```
`ts
```

```
set PORT=3000 && node filesystem-server.js -d D:/Projects
```

```
`
```

Note: By default, the service will run **8090** port.

Now, just mapping the **ajaxSettings** property of the FileManager component to the appropriate file operation methods in the filesystem-server.js file will allow to manage the physical file system with NodeJS file system provider.

```
`ts
```

```
let hostUrl = 'http://localhost:{port}';
```

```
// Initializing File Manager with NodeJS service.
```

```
let filemanagerInstance: FileManager = new FileManager({
```

```
// Replace the hosted port number in the place of "{port}"
```

```
  ajaxSettings: {
```

```
    url: hostUrl,
```

```
    downloadUrl: hostUrl + "Download",
```

```
    uploadUrl: hostUrl + "Upload",
```

```
    getImageUrl: hostUrl + "GetImage"
```

```
  }
```



```
});
filemanagerInstance.appendTo('#filemanager');
`
```

Note: To learn more about the file actions that can be performed with NodeJS file system provider, refer to this [link](#)

Google Drive file system provider

In ASP.NET Core, Google Drive file system provider in ASP.NET Core allows the users to manage the files and folders in a Google Drive account. The Google Drive file system provider works on ID basis where each file and folder have a unique ID. To get started, clone the [google-drive-aspcore-file-provider](#) using the following command.

```
`ts
git clone https://github.com/SyncfusionExamples/google-drive-aspcore-file-provider google-drive-aspcore-file-provider
cd google-drive-aspcore-file-provider
`
```

After generating the client secret data, copy the JSON data to the following specified JSON files in the cloned location.

- EJ2GoogleDriveFileProvider > credentials > client_secret.json
- GoogleOAuth2.0Base > credentials > client_secret.json

After updating the credentials, just build and run the project. Now, the project will be hosted in <http://localhost:{port}>, and it will ask to log on to the Gmail account created for the client secret credentials. Then, provide permission to access the Google Drive files by clicking the allow access button in the page. Now, just mapping the **ajaxSettings** property of the FileManager component to the appropriate controller methods will allow to manage the files from the Google Drive.

```
`ts
let hostUrl = 'http://localhost:{port}/';
// Initializing File Manager with Google drive service configuration.
let filemanagerInstance: FileManager = new FileManager({
  ajaxSettings: {
    // Replace the hosted port number in the place of "{port}"
    url: hostUrl + "api/GoogleDriveProvider/GoogleDriveFileOperations",
    downloadUrl: hostUrl + "api/GoogleDriveProvider/GoogleDriveDownload",
    uploadUrl: hostUrl + "api/GoogleDriveProvider/GoogleDriveUpload",
    getImageUrl: hostUrl + "api/GoogleDriveProvider/GoogleDriveGetImage"
  }
});
```

```
filemanagerInstance.appendTo('#filemanager');
```

,

Note: To learn more about the file actions that can be performed with Google drive file system provider, refer to this [link](#)

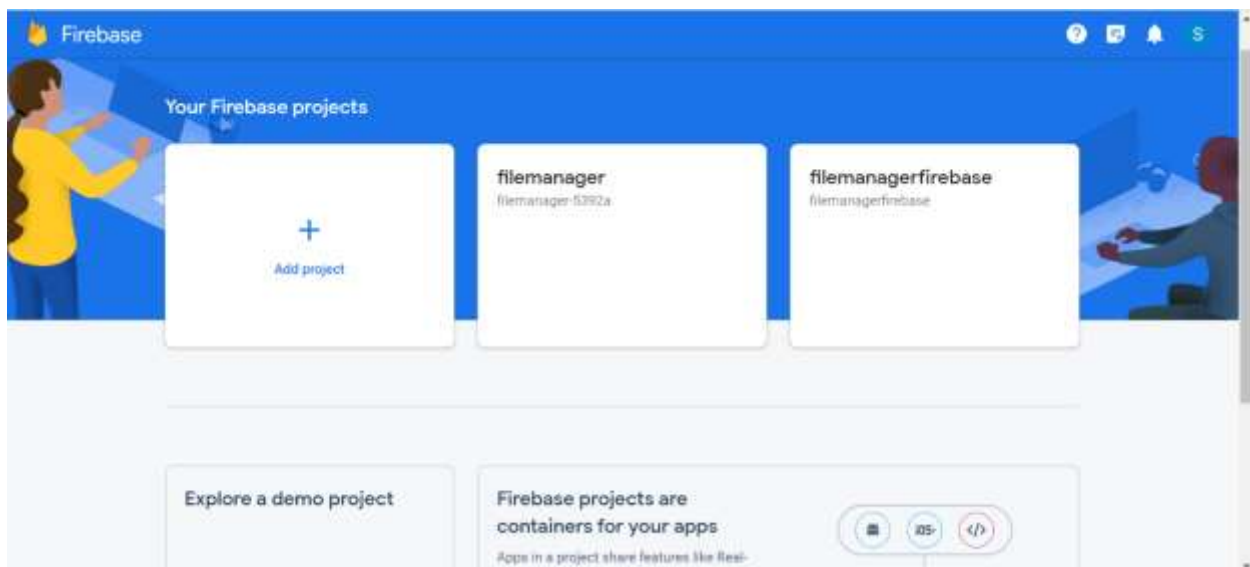
Firebase Realtime Database file system provider

The [Firebase Realtime Database](#) file system provider in **ASP.NET Core** provides the efficient way to store the File Manager file system in a cloud database as JSON representation.

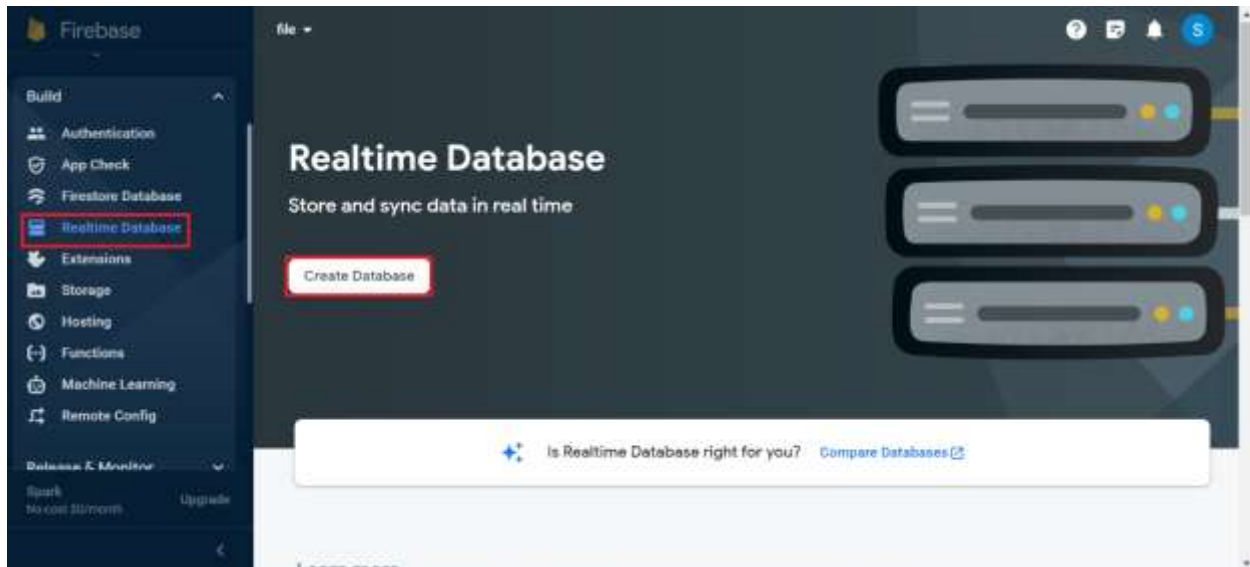
Generate Secret access key from service account

Follow the given steps to generate the secret access key:

- To access the Firebase console, please click on this [link](#). Once you have accessed the console, you can create a new project by filling in the necessary fields and clicking on the relevant buttons.



- Within the Firebase console, navigate to the **Build** tab. Under this tab, select the option for **Realtime Database**. From there, you can create a new database by clicking on the **Create Database** button.



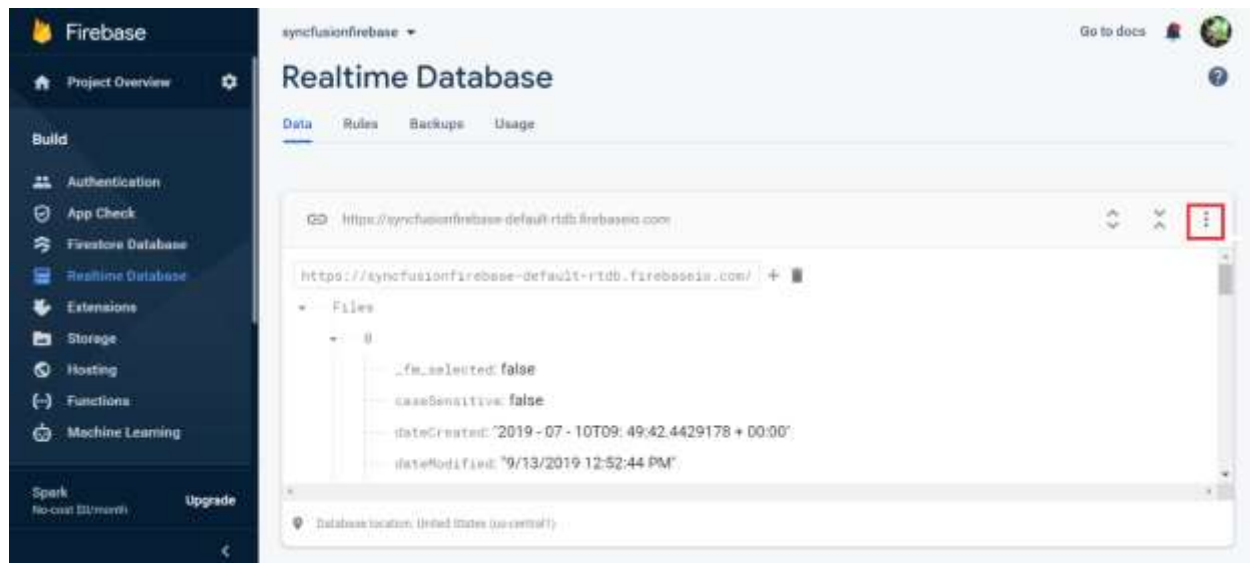
- To get started, create a root node and add any desired children to it. Please refer to the following code snippet for guidance on the structure of the JSON:

```
{
  "Files" : [ {
    "caseSensitive" : false,
    "dateCreated" : "8/22/2019 5:17:55 PM",
    "dateModified" : "8/22/2019 5:17:55 PM",
    "filterId" : "0/",
    "filterPath" : "/",
    "hasChild" : false,
    "id" : "5",
    "isFile" : false,
    "isRoot" : true,
    "name" : "Music",
    "parentId" : "0",
    "selected" : false,
    "showHiddenItems" : false,
    "size" : 0,
    "type" : "folder"
  },
  {
```

```
"caseSensitive" : false,
"dateCreated" : "8/22/2019 5:18:03 PM",
"dateModified" : "8/22/2019 5:18:03 PM",
"filterId" : "0/",
"filterPath" : "/",
"hasChild" : false,
"id" : "6",
"isFile" : false,
"isRoot" : true,
"name" : "videos",
"parentId" : "0",
"selected" : false,
"showHiddenItems" : false,
"size" : 0,
"type" : ""
}}
}
`
```

Here, the `Files` denotes the `rootNode` and the subsequent object refers to the children of the root node. `rootNode` will be taken as the root folder of the file system loaded which will be loaded in File Manager component.

- To import a JSON file into the Firebase Realtime Database, navigate to the **Data** tab and click on the action icon shown in the accompanying image. From there, select the **Import JSON** option and upload the JSON file that was created using the code provided above.

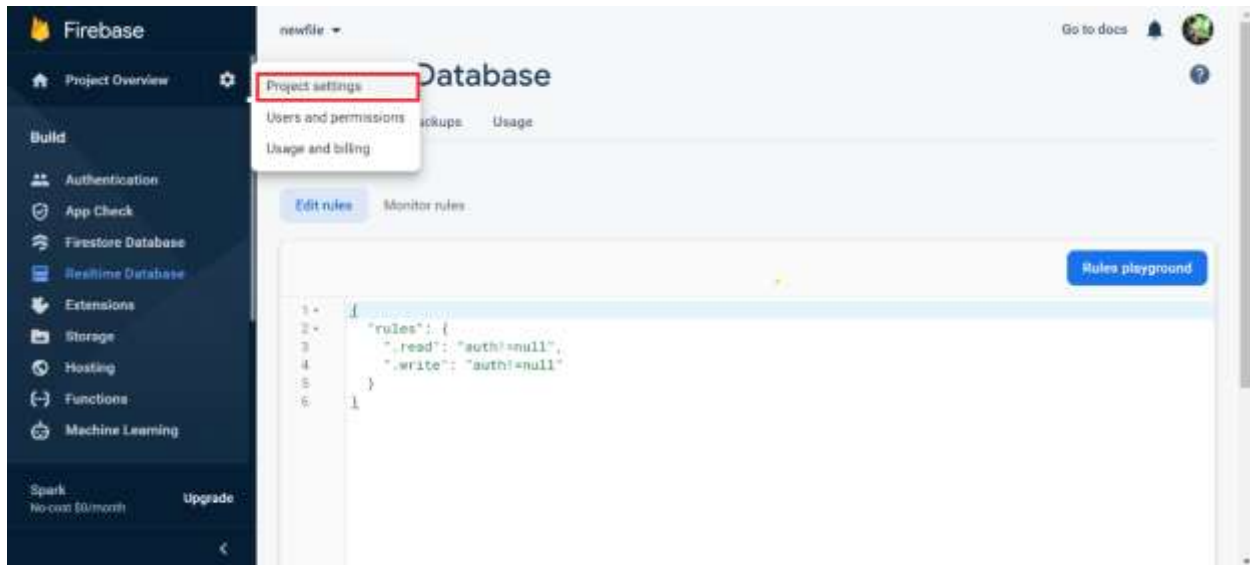


- To interact with the Firebase Realtime Database through your application, it is necessary to grant read and write permissions by defining appropriate rules in the Firebase project's **Rules tab**, as shown in the following code snippet. Once you have specified the rules, you can publish them by clicking the **Publish** button to enable the necessary authentication.

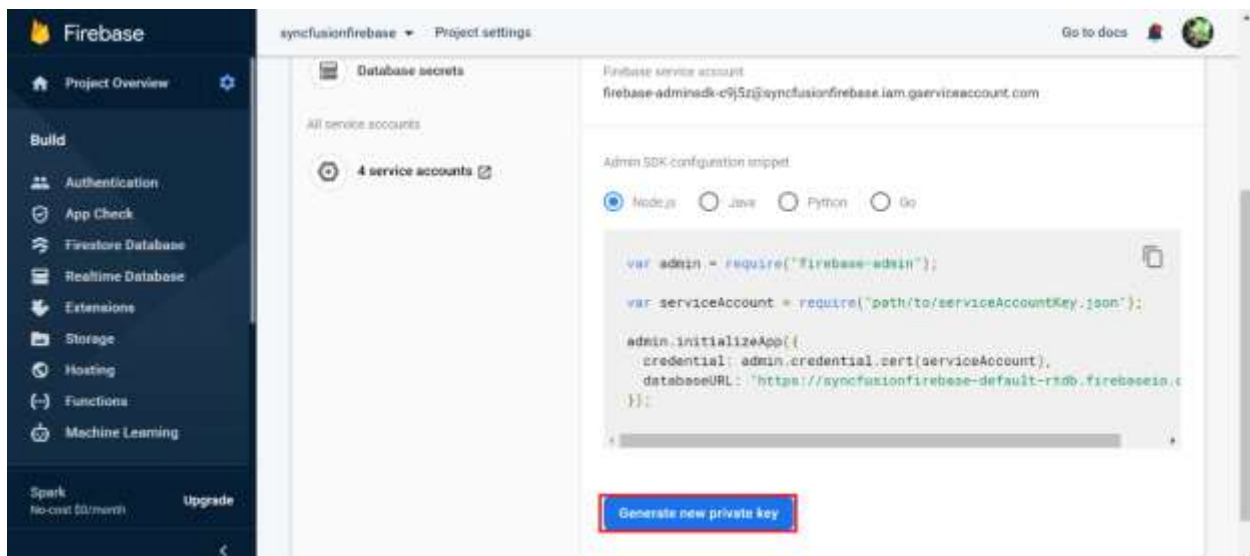
```
{
/ Visit https://firebase.google.com/docs/database/security to learn more about security rules. /
"rules": {
".read": "auth!=null",
".write": "auth!=null"
}
}
```

Note: By default, rules of a Firebase project will be **false**. To read and write the data, configure the **Rules** as given in the following code snippet in the *Rules* tab in the Firebase Realtime Database project.

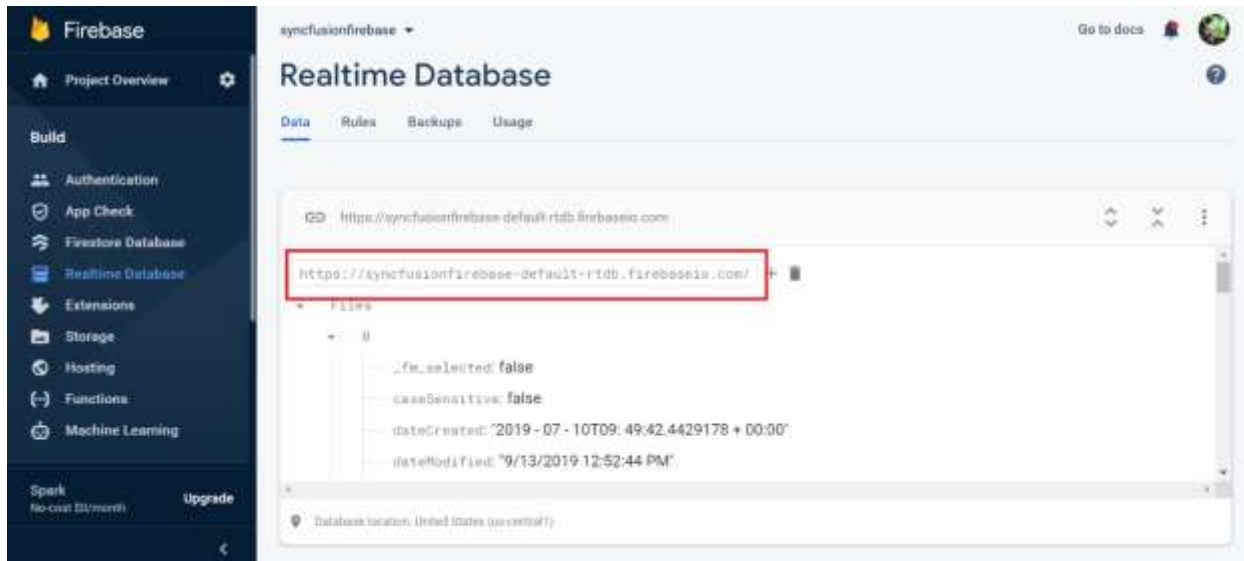
- Navigate to the project settings as instructed and then click on the **Service Account** tab.



- To obtain the access key JSON file, simply click on the **Generate new private key** button and then confirm by clicking the **Generate key** button in the pop-up window that appears.



- Next, you will need to clone the [firebase-realtime-database-apscore-file-provider](#) repository. Once cloned, simply open the project in Visual Studio and restore the NuGet package.
- Once you have generated the secret key, you will need to replace the JSON in the `access_key.json` file in the Firebase Realtime Database provider project with the newly generated key. This will enable authentication and allow you to perform read and write operations.
- In the **Data** tab, locate the project API URL and then paste it into the below mentioned section.



Register the Firebase Realtime Database by assigning *Firebase Realtime Database REST API link*, *rootNode*, and *serviceAccountKeyPath* parameters in the `RegisterFirebaseRealtimeDB` method of class `FirebaseRealtimeDBFileProvider` in the controller part of the ASP.NET Core application.

```
this.operation.RegisterFirebaseRealtimeDB(string apiUrl, string rootNode, string serviceAccountKeyPath)
、
```

Example:

```
this.operation.RegisterFirebaseRealtimeDB("{copy your API URL here}", "Files",
hostingEnvironment.ContentRootPath + "\\FirebaseRealtimeDBHelper\\access_key.json");
、
```

In the above code,

- `{copy your API URL here}` denotes Firebase Realtime Database REST API link.
- `Files` denotes newly created root node in Firebase Realtime Database.
- `hostingEnvironment.ContentRootPath + "\\FirebaseRealtimeDBHelper\\access_key.json` denotes service account key path which has authentication key for the Firebase Realtime Database data.

After configuring the Firebase Realtime Database service link, build and run the project. Now, the project will be hosted in `http://localhost:{port}` and just mapping the **ajaxSettings** property of the File Manager component to the appropriate controller methods allows to manage the files in the Firebase Realtime Database.

```
import { Component } from '@angular/core';
@Component({
selector: 'app-root',
styleUrls: ['app/app.component.css'],
templateUrl: 'app/app.component.html'
```

```

})
export class AppComponent {
  public ajaxSettings: object;
  public hostUrl: string = 'http://localhost:{port}/';
  public ngOnInit(): void {
    // Initializing File Manager with Firebase Realtime Database service.
    this.ajaxSettings = {
      // Replace the hosted port number in the place of "{port}"
      url = this.hostUrl + "api/FirebaseProvider/FirebaseRealtimeFileOperations",
      downloadUrl = this.hostUrl + "api/FirebaseProvider/FirebaseRealtimeDownload",
      uploadUrl = this.hostUrl + "api/FirebaseProvider/FirebaseRealtimeUpload",
      getImageUrl = this.hostUrl + "api/FirebaseProvider/FirebaseRealtimeGetImage"
    };
  }
}

```

> **Note:** To learn more about the file actions that can be performed with Firebase Realtime Database file system provider, refer to this [link](#)

IBM Cloud Object Storage file provider

The IBM Cloud Object Storage file provider module allows you work with the IBM Cloud Object Storage. It also provides the methods for performing various file actions such as creating a new folder, renaming files, and deleting files. The IBM Cloud Object Storage file provider serves the file provider support for the File Manager component with the IBM Cloud Object Storage. We can make use of IBM Cloud Object Storage file provider by installing the [ej2-filemanager-ibm-cos-node-file-provider](#) npm package or by cloning the [file provider](#) from the GitHub.

Using ej2-filemanager-ibm-cos-node-file-provider npm package

- Install the ej2-filemanager-ibm-cos-node-file-provider npm package by running the below command.

`ts

```
npm install @syncfusion/ej2-filemanager-ibm-cos-node-file-provider
```

- After installing the package, navigate to the ej2-filemanager-ibm-cos-node-file-provider package folder within the node-modules.
- Run the **npm install** command to install the dependent packages for file provider.

Cloning the filemanager-ibm-cos-node-file-provider from GitHub

- Clone the filemanager-ibm-cos-node-file-provider using the following command.

```
`ts
git clone https://github.com/SyncfusionExamples/filemanager-ibm-cos-node-file-provider.git
`
```

- After cloning, open the root folder and run the command **npm install** command.

To set the port in which the project to be hosted. Run the following command.

```
`ts
set PORT=3000 && node index.js
`
```

Note: By default, the service will run 8090 port.

Now, just mapping the **ajaxSettings** property of the FileManager component to the appropriate file operation methods in the index.js file will allow to manage the IBM Cloud Object Storage.

```
`ts
let hostUrl = 'http://localhost:{port}/';
// Initializing File Manager with IBM COS service.
let filemanagerInstance: FileManager = new FileManager({
// Replace the hosted port number in the place of "{port}"
ajaxSettings: {
url: hostUrl,
downloadUrl: hostUrl + "Download",
uploadUrl: hostUrl + "Upload",
getImageUrl: hostUrl + "GetImage"
}
});
filemanagerInstance.appendTo('#filemanager');
`
```

Note: To learn more about the file actions that can be performed with IBM Cloud Object Storage file provider, refer to this [link](#)

Localization in ##Platform_Name## File manager control

The file manager can be localized to any culture by defining the texts and messages of the file manager in the corresponding culture. The default locale of the file manager is **en** (English). The following table represents the default texts and messages of the file manager in **en** culture.

KEY	Text/Message
NewFolder	New folder
Upload	Upload
Delete	Delete
Rename	Rename
Download	Download
Cut	Cut
Copy	Copy
Paste	Paste
SortBy	Sort by
Refresh	Refresh
Item-Selection	item selected
Items-Selection	items selected
View	View
Details	Details
SelectAll	Select all
Open	Open
Tooltip-NewFolder	New folder
Tooltip-Upload	Upload
Tooltip-Delete	Delete
Tooltip-Rename	Rename
Tooltip-Download	Download
Tooltip-Cut	Cut
Tooltip-Copy	Copy
Tooltip-Paste	Paste
Tooltip-SortBy	Sort by
Tooltip-Refresh	Refresh
Tooltip-Selection	Clear selection
Tooltip-View	View
Tooltip-Details	Details
Tooltip-SelectAll	Select all
Name	Name

Size	Size
DateModified	Modified
DateCreated	Date created
Path	Path
Created	Created
Modified	Modified
Location	Location
Type	Type
Permission	Permission
Ascending	Ascending
Descending	Descending
None	None
View-LargeIcons	Large icons
View-Details	Details
Search	Search
Button-Ok	OK
Button-Cancel	Cancel
Button-Yes	Yes
Button-No	No
Button-Create	Create
Button-Save	Save
Header-NewFolder	Folder
Content-NewFolder	Enter your folder name
Header-Rename	Rename
Content-Rename	Enter your new name
Header-Rename-Confirmation	Rename Confirmation
Content-Rename-Confirmation	If you change a file name extension
Are you sure you want to change it?	
Header-Delete	Delete File
Content-Delete	Are you sure you want to delete this file?
Header-Multiple-Delete	Delete Multiple Files
Content-Multiple-Delete	Are you sure you want to delete these {0} files?
Header-Folder-Delete	Delete Folder

|Content-Folder-Delete|Are you sure you want to delete this folder?|

|Header-Duplicate|File exists|

|Content-Duplicate| already exists. Are you sure you want to replace it?|

|Header-Upload|Upload Files|

|Error|Error|

|Validation-Empty|The file or folder name cannot be empty.|

|Validation-Invalid|The file or folder name {0} contains invalid characters. Please use a different name. Valid file or folder names cannot end with a dot or space, and cannot contain any of the following characters: \\/:*?"<>\\||

|Validation-NewFolder-Exists|A file or folder with the name {0} already exists.|

|Validation-Rename-Exists|Cannot rename {0} to {1}| destination already exists.|

|Folder-Empty|This folder is empty|

|File-Upload|Drag files here to upload|

|Search-Empty|No results found|

|Search-Key|Try with different keywords|

|Filter-Empty|No results found|

|Filter-Key|Try with different filter|

|Sub-Folder-Error|The destination folder is the subfolder of the source folder|

|Same-Folder-Error|The destination folder is the same as the source folder.|

|Access-Denied|Access Denied|

|Access-Details|You don't have permission to access this folder|

|Header-Retry|File Already Exists|

|Content-Retry|A file with this name already exists in this folder. What would you like to do?|

|Button-Keep-Both|Keep both|

|Button-Replace|Replace|

|Button-Skip|Skip|

|ApplyAll-Label|Do this for all current items|

|KB|KB|

|Access-Message|{0} is not accessible. You need permission to perform the {1} action.|

|Network-Error|NetworkError: Failed to send on XMLHttpRequest: Failed to load|

|Server-Error|ServerError: Invalid response from|

The below example shows adding the German culture locale(**de-DE**)

INDEX.JS

```
//Defining texts and messages corresponding to German culture
```

```

ej.base.L10n.load({
  'de': {
    'filemanager': {
      "NewFolder": "Neuer Ordner",
      "Upload": "Hochladen",
      "Delete": "Löschen",
      "Rename": "Umbenennen",
      "Download": "Herunterladen",
      "Cut": "Schnitt",
      "Copy": "Kopieren",
      "Paste": "Einfügen",
      "SortBy": "Sortiere nach",
      "Refresh": "Aktualisierung",
      "Item-Selection": "Artikel ausgewählt",
      "Items-Selection": "Elemente ausgewählt",
      "View": "Aussicht",
      "Details": "Einzelheiten",
      "SelectAll": "Wählen Sie Alle",
      "Open": "Öffnen",
      "Tooltip-NewFolder": "Neuer Ordner",
      "Tooltip-Upload": "Hochladen",
      "Tooltip-Delete": "Löschen",
      "Tooltip-Rename": "Umbenennen",
      "Tooltip-Download": "Herunterladen",
      "Tooltip-Cut": "Schnitt",
      "Tooltip-Copy": "Kopieren",
      "Tooltip-Paste": "Einfügen",
      "Tooltip-SortBy": "Sortiere nach",
      "Tooltip-Refresh": "Aktualisierung",
      "Tooltip-Selection": "Auswahl aufheben",
      "Tooltip-View": "Aussicht",
      "Tooltip-Details": "Einzelheiten",
      "Tooltip-SelectAll": "Wählen Sie Alle",
      "Name": "Name",
      "Size": "Größe",
      "DateModified": "Geändert",
      "DateCreated": "Datum erstellt",
      "Path": "Pfad",
      "Modified": "Geändert",
      "Created": "Erstellt",
      "Location": "Ort",
      "Type": "Art",
      "Permission": "Genehmigung",
      "Ascending": "Aufsteigend",
      "Descending": "Absteigend",
      "None": "Keiner",
      "View-LargeIcons": "Große Icons",
      "View-Details": "Einzelheiten",
      "Search": "Suche",
      "Button-Ok": "OK",
      "Button-Cancel": "Stornieren",
      "Button-Yes": "Ja",
      "Button-No": "Nein",
      "Button-Create": "Erstellen",
      "Button-Save": "Sparen",
      "Header-NewFolder": "Mappe",
      "Content-NewFolder": "Geben Sie Ihren Ordnernamen ein",
    }
  }
});

```

```

        "Header-Rename": "Umbenennen",
        "Content-Rename": "Geben Sie Ihren neuen Namen ein",
        "Header-Rename-Confirmation": "Bestätigung umbenennen",
        "Content-Rename-Confirmation": "Wenn Sie eine
Dateinamenerweiterung ändern, wird die Datei möglicherweise instabil.
Möchten Sie sie wirklich ändern?",
        "Header-Delete": "Datei löschen",
        "Content-Delete": "Möchten Sie diese Datei wirklich löschen?",
        "Header-Multiple-Delete": "Mehrere Dateien löschen",
        "Content-Multiple-Delete": "Möchten Sie diese {0} Dateien
wirklich löschen?",
        "Header-Folder-Delete": "Lösche Ordner",
        "Content-Folder-Delete": "Möchten Sie diesen Ordner wirklich
löschen?",
        "Header-Duplicate": "Datei / Ordner existiert",
        "Content-Duplicate": "{0} existiert bereits. Möchten Sie
umbenennen und einfügen?",
        "Header-Upload": "Daten hochladen",
        "Error": "Error",
        "Validation-Empty": "Der Datei - oder Ordnername darf nicht leer
sein.",
        "Validation-Invalid": "Der Datei- oder Ordnername {0} enthält
ungültige Zeichen. Bitte verwenden Sie einen anderen Namen. Gültige Datei-
oder Ordnernamen dürfen nicht mit einem Punkt oder Leerzeichen enden und
keines der folgenden Zeichen enthalten: \\ /: *? \" < > | ",
        "Validation-NewFolder-Exists": "Eine Datei oder ein Ordner mit
dem Namen {0} existiert bereits.",
        "Validation-Rename-Exists": "{0} kann nicht in {1} umbenannt
werden: Ziel existiert bereits.",
        "Folder-Empty": "Dieser Ordner ist leer",
        "File-Upload": "Dateien zum Hochladen hierher ziehen",
        "Search-Empty": "Keine Ergebnisse gefunden",
        "Search-Key": "Versuchen Sie es mit anderen Stichwörtern",
        "Filter-Empty": "keine Ergebnisse gefunden",
        "Filter-Key" : "Versuchen Sie es mit einem anderen Filter",
        "Sub-Folder-Error": "Der Zielordner ist der Unterordner des
Quellordners.",
        "Same-Folder-Error": "Der Zielordner ist derselbe wie der
Quellordner.",
        "Access-Denied": "Zugriff verweigert",
        "Access-Details": "Sie haben keine Berechtigung, auf diesen
Ordner zuzugreifen.",
        "Header-Retry": "Die Datei existiert bereits",
        "Content-Retry": "In diesem Ordner ist bereits eine Datei mit
diesem Namen vorhanden. Was möchten Sie tun?",
        "Button-Keep-Both": "Behalte beides",
        "Button-Replace": "Ersetzen",
        "Button-Skip": "Überspringen",
        "ApplyAll-Label": "Mache das für alle aktuellen Artikel",
        "KB": "KB",
        "Access-Message": "{0} ist nicht zugänglich. Sie benötigen die
Berechtigung, um die Aktion {1} auszuführen.",
        "Network-Error": "NetworkError: Fehler beim Senden auf
XMLHttpRequest: Fehler beim Laden",
        "Server-Error": "ServerError: Ungültige Antwort von"
    }
}

```

```

}))
var hostUrl = 'https://ej2-aspcore-service.azurewebsites.net/';
// inject feature modules of the file manager
ej.filemanager.FileManager.Inject(ej.filemanager.DetailsView,ej.filemanager.
Toolbar,ej.filemanager.NavigationPane);
// initialize File Manager component
var filemanagerInstance = new ej.filemanager.FileManager({
    ajaxSettings: {
        url: hostUrl + 'api/FileManager/FileOperations',
        getImageUrl: hostUrl + 'api/FileManager/GetImage',
        uploadUrl: hostUrl + 'api/FileManager/Upload',
        downloadUrl: hostUrl + 'api/FileManager/Download'
    },
    //defining the locale for File Manager
    locale: 'de'
});
// render initialized File Manager
filemanagerInstance.appendTo('#filemanager');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 File Manager</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Essential JS 2 File Manager
Component">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
filemanager/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```

```

<div id="filemanager"></div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Virtualization in ##Platform_Name## File manager control

File Manager's UI virtualization allows you for the dynamic loading of a large number of directories and files in both the detailsView and largeIconsView without degrading its performance.

Module Injection

To use UI virtualization, you must import the `virtualization` module from the `ej2-filemanager` package and inject it using the `FileManager.Inject()` function.

```
`ts
```

```
import { FileManager, Virtualization } from '@syncfusion/ej2-filemanager';
```

```
FileManager.Inject(Virtualization);
```

```
,
```

Enable Virtualization

In order to enable `virtualization`, you must set the `enableVirtualization` property to true.

In the instance below, a sizable collection of files can be found in the folders **Documents** and **Text Documents**.

INDEX.JS

```

var hostUrl = 'https://ej2-aspcore-service.azurewebsites.net/';
// inject feature modules of the file manager
ej.filemanager.FileManager.Inject(ej.filemanager.DetailsView,ej.filemanager.Toolbar,ej.filemanager.NavigationPane,ej.filemanager.enableVirtualization);
// initialize File Manager component
var filemanagerInstance = new ej.filemanager.FileManager({
    ajaxSettings: {
        url: hostUrl + 'api/FileManager/FileOperations',
        getImageUrl: hostUrl + 'api/FileManager/GetImage',
        uploadUrl: hostUrl + 'api/FileManager/Upload',
        downloadUrl: hostUrl + 'api/FileManager/Download'
    },
    view: 'Details',
    enableVirtualization: true,
    beforeSend: function(args) {
        args.ajaxSettings.beforeSend = function (args) {
            args.httpRequest.setRequestHeader('Authorization',
'FileBrowser');
        };
    },
    beforeImageLoad: function(args) {
        args.imageUrl = args.imageUrl + '&rootName=' + 'FileBrowser';
    },

```



```

        beforeDownload: function(args) {
            args.data.rootFolderName = 'FileBrowser';
        },
    });
    // render initialized File Manager
    filemanagerInstance.appendTo('#filemanager');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 File Manager</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Essential JS 2 File Manager
Component">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
filemanager/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="filemanager"></div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

Limitations for Virtualization

- Programmatic selection using the **selectAll** method is not supported with virtual scrolling.
- The keyboard shortcut **CTRL+A** will only select the files and directories that are currently visible within the viewport, rather than selecting all files and directories in the entire directory tree.
- Selected file items are not maintained while scrolling, considering the performance of the component.

Accessibility in ##Platform_Name## File Manager component

The File Manager component followed the accessibility guidelines and standards, including [ADA](#), [Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the File Manager component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support | |

| [Section 508](#) Support | |

| Screen Reader Support | |

| Right-To-Left Support | |

| Color Contrast | |

| Mobile Device Support | |

| Keyboard Navigation Support | |

| [Accessibility Checker](#) Validation | |

| [Axe-core](#) Accessibility Validation | |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<div> - All features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>

WAI-ARIA attributes

The File Manager component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the File Manager component:

| Attributes | Purpose |

| --- | --- |

| **role** | Used to convey a significant and contextual message to the user. |

| **aria-disabled** | Indicates whether the File Manager component is in disabled state. |

| **aria-haspopup** | Indicates whether the toolbar item has a popup list or not. |

| **aria-orientation** | Indicates whether the File Manager element is oriented horizontally or vertically. |

| **aria-expanded** | Indicates whether the Treeview node has been expanded. |

| **aria-owns** | Contains the ID of the suggestion list to indicate popup as a child element. |

| **aria-activedescendent** | Holds the ID of the active list item to focus its descendant child element. |

| **aria-level** | Specifies the level of the element in Treeview Structure. |

| **aria-selected** | Indicates whether a particular node is in selected state. |

| **aria-placeholder** | Represents a hint (word or phrase) to the user about what to enter in the text field. |

| **aria-label** | Provides an accessible name for the element. |

| **aria-checked** | Indicates whether the checkbox is in checked state. |

| **aria-labelledby** | Provides a label for the dialog. Typically, the "aria-labelledby" attribute will contain the id of the element used as the dialog's title. |

| **aria-describedby** | This attribute points to the Dialog element describing the one it's set on. |

| **aria-modal** | Indicates whether an element is a modal when display. |

| **aria-colcount** | Specifies the number of columns in full table. |

| **aria-colindexnt** | Defines the number of columns within a table in details view. |

| **aria-rowspan** | Defines the number of rows a cell spanned within a table in details view. |

| **aria-colspan** | Defines the number of columns a cell spanned within a table in details view. |

| **aria-sort** | Indicates whether items in the table are sorted in ascending or descending order. |

| **aria-grabbed** | When the folder/file item is chosen for dragging, the aria-grabbed attribute is set to "true." If it's set to "false," the element can be grabbed for drag-and-drop, but it won't be actively held.
|

| **aria-busy** | This attribute is set to false when table content is loaded. |

| **aria-multiselectable** | Defines more than one item has been selected. |

Keyboard interaction

The File Manager component followed the **keyboard interaction** guidelines, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the File Manager component.

| **Press** | **To do this** |

| --- | --- |

| **Page Down** | Scrolls down to the next folder or file and selects the first item when files are loaded. |

| **Page Up** | Scrolls up to previous folder and select the first item when files are loaded. |

| **Enter** | Selects the focused item and navigate through the child elements. |

| **Tab** | Focuses on the first element of toolbar and navigates through the next tab indexed element. |

| **Esc(Escape)** | Closes the image when it is in open state. |

| **Alt+N** | Creates a new folder dialog. |

| **F5** | Refresh the file manager element. |

| **Home** | Navigate through the first element of details view or large icons view. |

| **End** | Navigate through the last element of details view or large icons view. |

| **Move Left** | Scrolls left to the previous folder and select the first item when files are loaded |

| **Move Right** | Scrolls right to the previous folder and select the first item when files are loaded
|

| **Alt+Enter** | Shows the get details info for selected folder. |

| **Shift+Right** | Allows multiselection. Select the file or folder at the right of the previously selected folder. |

| **Shift+Left** | Allows multiselection. Select the file or folder at the left of the previously selected folder. |

| **Shift+Down** | Allows multiselection. Select the file or folder till the focused index. |

| **Shift+Delete** | Permanently deletes the selected file or folder in the file manager element. |

| **Delete** | Deletes the selected file or folder in the file manager element. |

| **Shift+Up** | Allows multiselection. Select the file or folder till the focused index. |

Ctrl+C	Copies the selected file or folder in the file manager element.	
Ctrl+V	Pastes the copied/cut file or folder in the file manager element.	
Ctrl+X	Cuts the selected file or folder in the file manager element.	
Ctrl+A	Select all the files or folders in the details view or large icons view.	
F2	Creates a rename dialog for a selected file or folder in the file manager element.	
Shift+F10	Opens the context menu for the selected file or folder in the file manager element.	
Ctrl+D	Downloads the list of selected files or folders in the file manager element.	
Ctrl+Shift+1	Changes the file manager layout to details view.	
Ctrl+Shift+2	Changes the file manager layout to details view.	

Ensuring accessibility

The File Manager component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the File Manager component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the File Manager component with accessibility tools.

See also

- [Accessibility in Syncfusion ##Platform_Name## components](#)

Access control in ##Platform_Name## File manager control

The FileManager allows you to define access permissions for folders and files using a set of access rules to user(s).

- [Access Rules](#)
- [Permissions](#)

Access Rules

The FileAccessController allows you to define security permissions for folders and files using a set of folder or file access rules.

To set up access rules for folders (including their files and sub-folders) and individual files, use the SetRules() method in the controller. The following table represents the AccessRule properties available for file and folder:

Properties	Applicable for file	Applicable for folder	Description
---	---	---	---
Copy	Yes	Yes	Allows access to copy a file or folder.
Read	Yes	Yes	Allows access to read a file or folder.
Write	Yes	Yes	Allows permission to write a file or folder.

WriteContents	No	Yes	Allows permission to write the content of folder.
Download	Yes	Yes	Allows permission to download a file or folder.
Upload	No	Yes	Allows permission to upload to the folder.
Path	Yes	Yes	Specifies the path to apply the rules, which are defined.
Role	Yes	Yes	Specifies the role to which the rule is applied.
IsFile	Yes	Yes	Specifies whether the rule is specified for folder or file.

The following syntax represents the access Rules for Administrator using file or folder.

```
`ts
```

```
//Adminstrator
```

```
//Access Rules for File
```

```
new AccessRule { Path = "/", Role = "Administrator", Read = Permission.Allow, Write =  
Permission.Allow, Copy = Permission.Allow, Download = Permission.Allow, IsFile = true },
```

```
// Access Rules for folder
```

```
new AccessRule { Path = "*", Role = "Administrator", Read = Permission.Allow, Write = Permission.Allow,  
Copy = Permission.Allow, WriteContents = Permission.Allow, Upload = Permission.Allow, Download =  
Permission.Deny, IsFile = false },
```

```
,
```

The following syntax represent the access Rules for Default user using file or folder.

```
`ts
```

```
//Default User
```

```
//Access Rules for File
```

```
new AccessRule { Path = "/", Role = "Default User", Read = Permission.Deny, Write = Permission.Deny,  
Copy = Permission.Deny, Download = Permission.Deny, IsFile = true },
```

```
// Access Rules for folder
```

```
new AccessRule { Path = "*", Role = "Default User", Read = Permission.Deny, Write = Permission.Deny,  
Copy = Permission.Deny, WriteContents = Permission.Deny, Upload = Permission.Deny, Download =  
Permission.Deny, IsFile = false },
```

```
,
```

Permissions

It helps to explain how to apply security permission to file manager file or folder using access rules. The following table represent the value that determines the permission.

Value	Description
-------	-------------

---	---
-----	-----

Allow	Allows you to do read, write, copy, and download operations.
-------	--

Deny	Denies you to do read, write, copy, and download operations.
------	--

Use the **Role** property to apply created roles to the file manager. After that, the file manager displays folder or file and allow permission based on assigned roles.

The following syntax represent how to apply permission based on assigned roles

Permission denied for administrator to write a file or folder.

```
`ts
// For file
new AccessRule { Path = "/", Role = "Administrator", Read = Permission.Allow, Write = Permission.Deny,
IsFile = true},
// For folder
new AccessRule { Path = "*", Role = "Administrator", Read = Permission.Allow, Write = Permission.Deny,
IsFile = false},
`
```

The following syntax represent how to allow or deny permission based on file or folder access rule.

Permission denied for writing except for particular file or folder.

```
`ts
// Deny writing for particular folder
new AccessRule { Path = "/Documents", Role = "Document Manager", Read = Permission.Allow, Write =
Permission.Deny, Copy = Permission.Allow, WriteContents = Permission.Deny, Upload =
Permission.Deny, Download = Permission.Deny, IsFile = false },
// Deny writing for particular file
new AccessRule { Path = "/Pictures/Employees/Adam.png", Role = "Document Manager", Read =
Permission.Allow, Write = Permission.Deny, Copy = Permission.Deny, Download = Permission.Deny,
IsFile = true },
`
```

Permission denied for writing and uploading in root folder.

```
`ts
// Folder Rule
new AccessRule { Path = "/", Role = "Document Manager", Read = Permission.Allow, Write =
Permission.Deny, Copy = Permission.Deny, WriteContents = Permission.Deny, Upload =
Permission.Deny, Download = Permission.Deny, IsFile = false },
`
```

The following example demonstrate the file manager rendered with access control support.

INDEX.JS

```
var hostUrl = 'https://ej2-aspcore-service.azurewebsites.net/';
// inject feature modules of the file manager
ej.filemanager.FileManager.Inject(ej.filemanager.DetailsView,ej.filemanager.
Toolbar,ej.filemanager.NavigationPane);
// initialize File Manager component
```

```

var filemanagerInstance = new ej.filemanager.FileManager({
  ajaxSettings: {
    url: hostUrl + 'api/FileManagerAccess/FileOperations',
    uploadUrl: hostUrl + 'api/FileManagerAccess/Upload',
    downloadUrl: hostUrl + 'api/FileManagerAccess/Download',
    getImageUrl: hostUrl + 'api/FileManagerAccess/GetImage'
  },
});
// render initialized File Manager
filemanagerInstance.appendTo('#filemanager');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 File Manager</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 File Manager
Component">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
filemanager/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="filemanager"></div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
<script src="index.js" type="text/javascript"></script>

```



```
</body></html>
```

How To

Adding custom item to context menu in `##Platform_Name##` File manager control

The context menu can be customized using the [contextMenuSettings](#), [menuOpen](#), and [menuClick](#) events.

The following example shows adding a custom item in the context menu.

The [contextMenuSettings](#) is used to add new menu item. The [menuOpen](#) event is used to add the icon to the new menu item. The [menuClick](#) event is used to add an event handler to the new menu item.

INDEX.JS

```
var hostUrl = 'https://ej2-aspcore-service.azurewebsites.net/';
// inject feature modules of the file manager
ej.filemanager.FileManager.Inject(ej.filemanager.DetailsView,ej.filemanager.
Toolbar,ej.filemanager.NavigationPane);
// initialize File Manager component
var filemanagerInstance = new ej.filemanager.FileManager({
  ajaxSettings: {
    url: hostUrl + 'api/FileManager/FileOperations',
    getImageUrl: hostUrl + 'api/FileManager/GetImage',
    uploadUrl: hostUrl + 'api/FileManager/Upload',
    downloadUrl: hostUrl + 'api/FileManager/Download'
  },
  // Custom menu item added to context menu
  contextMenuSettings: {
    file: ["Custom", "Open", "|", "Delete", "Download", "Rename", "|",
"Details"],
    folder: ["Custom", "Open", "|", "Delete", "Download", "Rename", "|",
"Details"],
    layout: ["Custom", "SortBy", "View", "Refresh", "|", "NewFolder",
"Upload", "|", "Details", "|", "SelectAll"],
    visible: true
  },
  menuOpen: menuOpen,
  menuClick: menuClick
});
// Icon added to custom menu item
function menuOpen(args) {
  for(let i = 0; i<args.items.length; i++) {
    if(args.items[i].id === this.element.id + '_cm_custom') {
      args.items[i].iconCss = 'e-icons e-fe-tick';
    }
  }
}
// event for custom menu item
function menuClick(args) {
  if (args.item.text === 'Custom') {
    alert('You have clicked custom menu item')
  }
}
// render initialized File Manager
filemanagerInstance.appendTo('#filemanager');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 File Manager</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 File Manager
Component">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
filemanager/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="filemanager"></div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Adding custom item to toolbar in ##Platform_Name## File manager control

You can modify the items displayed in the toolbar by utilizing the [toolbarItems](#) API. To display both default and customized items, it's essential to assign a unique **name** to each item. Additionally, you have the flexibility to alter the default items by adjusting properties such as **tooltipText**, **iconCss**, **Text**, **suffixIcon** and more. This level of customization allows you to tailor the toolbar to your specific requirements and design preferences. The names used in the code example below serve as unique identifiers for default toolbar items, while custom items can be assigned any unique name value to distinguish them from the defaults.

For instance, here's an example of how to add a custom checkbox to the toolbar using the **template** property. Here we have modified the default **New Folder** item and added a custom toolbar item for selection.

INDEX.JS

```
var hostUrl = 'https://ej2-aspcore-service.azurewebsites.net/';
// inject feature modules of the file manager
ej.filemanager.FileManager.Inject(ej.filemanager.DetailsView,ej.filemanager.
Toolbar,ej.filemanager.NavigationPane);
// initialize File Manager component
var filemanagerInstance = new ej.filemanager.FileManager({
  ajaxSettings: {
    url: hostUrl + 'api/FileManager/FileOperations',
    getImageUrl: hostUrl + 'api/FileManager/GetImage',
    uploadUrl: hostUrl + 'api/FileManager/Upload',
    downloadUrl: hostUrl + 'api/FileManager/Download'
  },
  //Custom item added along with default item
  toolbarItems: [{ text: 'Create folder' , name: 'NewFolder', prefixIcon:
'e-plus', tooltipText: 'Create folder' },
  { name: 'Upload' },
  { name: 'SortBy' },
  { name: 'Refresh' },
  { name: 'Cut' },
  { name: 'Copy' },
  { name: 'Paste' },
  { name: 'Delete' },
  { name: 'Download' },
  { name: 'Rename' },
  { template: '<input id="checkbox" type="checkbox"/>', name: 'Select' },
  { name: 'Selection' },
  { name: 'View' },
  { name: 'Details' }]
});
// render initialized File Manager
filemanagerInstance.appendTo('#filemanager');
// Render Checkbox in template
var checkbox = new ej.buttons.CheckBox({ label: 'Select All', checked:
false, change: onChange }, '#checkbox');
// on checkbox change select all or clear selection
function onChange(args) {
  if (args.checked) {
    filemanagerInstance.selectAll();
    checkbox.label = 'Unselect All';
  }
  else {
    filemanagerInstance.clearSelection();
    checkbox.label = 'Select All';
  }
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 File Manager</title>
```

```

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Essential JS 2 File Manager
Component">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
filemanager/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="filemanager"></div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Enable/disable toolbar item in ##Platform_Name## File manager control

The toolbar items can be enabled/disabled by specifying the items in [enableToolbarItems](#) or [disableToolbarItems](#) methods respectively.

The following example shows enabling and disabling toolbar items on button click.

INDEX.JS

```

var hostUrl = 'https://ej2-aspcore-service.azurewebsites.net/';
// inject feature modules of the file manager
ej.filemanager.FileManager.Inject(ej.filemanager.DetailsView, ej.filemanager.
Toolbar, ej.filemanager.NavigationPane);
// initialize File Manager component

```

```

var filemanagerInstance = new ej.filemanager.FileManager({
  ajaxSettings: {
    url: hostUrl + 'api/FileManager/FileOperations',
    getImageUrl: hostUrl + 'api/FileManager/GetImage',
    uploadUrl: hostUrl + 'api/FileManager/Upload',
    downloadUrl: hostUrl + 'api/FileManager/Download'
  },
  height: "330px"
});
// render initialized File Manager
filemanagerInstance.appendTo('#filemanager');
// Click event for enable button
document.getElementById("enable").onclick = function(args) {
  // Enable new folder toolbar item
  filemanagerInstance.enableToolbarItems(["newfolder"]);
}
// Click event for disable button
document.getElementById("disable").onclick = function(args) {
  // Disable new folder toolbar item
  filemanagerInstance.disableToolbarItems(["newfolder"]);
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 File Manager</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 File Manager
Component">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
filemanager/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>

```

```

</head>
<body>

    <div id="filemanager"></div>
    <button id="enable" class="e-btn e-success">Enable New Folder toolbar
item</button>
    <button id="disable" class="e-btn e-danger">Disable New Folder toolbar
item</button>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customize custom thumbnail in ##Platform_Name## File manager control

The default appearance of the file manager can customize with your own icon by using [showThumbnail](#) property.

The following example demonstrate how to add a custom icon in largeicons view.

INDEX.JS

```

var hostUrl = 'https://ej2-aspcore-service.azurewebsites.net/';
// inject feature modules of the file manager
ej.filemanager.FileManager.Inject(ej.filemanager.DetailsView,ej.filemanager.
Toolbar,ej.filemanager.NavigationPane);
// initialize File Manager component
var filemanagerInstance = new ej.filemanager.FileManager({
    ajaxSettings: {
        url: hostUrl + 'api/FileManager/FileOperations',
        getImageUrl: hostUrl + 'api/FileManager/GetImage',
        uploadUrl: hostUrl + 'api/FileManager/Upload',
        downloadUrl: hostUrl + 'api/FileManager/Download'
    },
    showThumbnail: false,
});
// render initialized File Manager
filemanagerInstance.appendTo('#filemanager');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 File Manager</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Essential JS 2 File Manager
Component">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
filemanager/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="filemanager"></div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Nested items in ##Platform_Name## File manager control

FileManager can be rendered inside the other components like Tab, Dialog, and more.

- [Adding file manager inside the dialog](#)
- [Adding file manager inside the tab](#)

Adding file manager inside the dialog

The following example shows the file manager component rendered inside the dialog. Click the browse button in the Uploader element to open the File Manager inside the Dialog control.

INDEX.JS

```

var hostUrl = 'https://ej2-aspcore-service.azurewebsites.net/';
// inject feature modules of the file manager
ej.filemanager.FileManager.Inject(ej.filemanager.DetailsView,ej.filemanager.
Toolbar,ej.filemanager.NavigationPane);
// Initialize the Uploader component
var uploadObject = new ej.inputs.Uploader({});
uploadObject.appendTo('#fileupload');
// Initialize the Button component
var btnObj = new ej.buttons.Button({});
btnObj.appendTo('#openBtn');

```

```

// Initialize the Dialog component
var dialogObj = new ej.popups.Dialog({
  header: 'Select a file',
  showCloseIcon: true,
  closeOnEscape: false,
  width: '850px',
  visible: false,
  target: document.getElementById('target'),
  animationSettings: { effect: 'None' },
  open: dialogOpen,
  close: dialogClose
});
dialogObj.appendTo('#dialog');
var contextmenuItems = ['Open', '|', 'Cut', 'Copy', 'Delete', 'Rename', '|',
'Details'];
// Initialize the FileManager component
var filemanagerInstance = new ej.filemanager.FileManager({
  ajaxSettings: {
    url: hostUrl + 'api/FileManager/FileOperations',
    getImageUrl: hostUrl + 'api/FileManager/GetImage',
    uploadUrl: hostUrl + 'api/FileManager/Upload',
    downloadUrl: hostUrl + 'api/FileManager/Download'
  },
  allowMultiSelection: false,
  toolbarSettings: {
    items: ['NewFolder', 'Upload', 'Delete', 'Cut', 'Copy', 'Rename',
'SortBy', 'Refresh', 'Selection', 'View', 'Details']],
  contextMenuSettings: {
    file: contextmenuItems,
    folder: contextmenuItems
  },
  fileOpen : onFileOpen
});
filemanagerInstance.appendTo('#filemanager');
document.getElementById('openBtn').onclick = function() {
  dialogObj.show();
  dialogOpen();
  filemanagerInstance.path = '/';
  filemanagerInstance.selectedItems = [];
  filemanagerInstance.refresh();
};
// 'Uploader' will be shown, if Dialog is closed
function dialogClose() {
  document.getElementById('container').style.display = 'block';
}
// 'Uploader' will be hidden, if Dialog is opened
function dialogOpen() {
  document.getElementById('container').style.display = 'none';
}
// File Manager's fileOpen event function
function onFileOpen(args) {
  var file = args.fileDetails;
  if (file.isFile) {
    args.cancel = true;
    if (file.size <= 0 ) { file.size = 10000; }
    uploadObject.files = [{name: file.name, size: file.size, type:
file.type }];
  }
}

```



```
        dialogObj.hide();  
    }  
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>  
    <title>Essential JS 2 File Manager</title>  
    <meta charset="utf-8">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <meta name="description" content="Essential JS 2 File Manager  
Component">  
    <meta name="author" content="Syncfusion">  
    <link href="index.css" rel="stylesheet">  
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-  
base/styles/material.css" rel="stylesheet">  
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-  
inputs/styles/material.css" rel="stylesheet">  
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-  
popups/styles/material.css" rel="stylesheet">  
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-  
buttons/styles/material.css" rel="stylesheet">  
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-  
splitbuttons/styles/material.css" rel="stylesheet">  
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-  
layouts/styles/material.css" rel="stylesheet">  
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-  
navigations/styles/material.css" rel="stylesheet">  
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-  
grids/styles/material.css" rel="stylesheet">  
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-  
filemanager/styles/material.css" rel="stylesheet">  
  
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"  
type="text/javascript"></script>  
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type  
="text/javascript"></script>  
</head>  
<body>  
  
    <div class="control-section">  
        <div id="container" class="fileupload">  
            <input type="file" id="fileupload" name="UploadFiles">  
            <button id="openBtn" class="dlgbtn"  
type="button">Browse...</button>  
        </div>  
        <div id="target" class="control-section">  
            <div id="dialog">  
                <div id="filemanager"></div>  
            </div>  
        </div>  
</div>  
<script>  
var ele = document.getElementById('container');  
if(ele) {
```

```

    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Adding file manager inside the tab

The following example demonstrate that the file manager component is placed inside the content area of tab element.

INDEX.JS

```

var hostUrl = 'https://ej2-aspcore-service.azurewebsites.net/';
// inject feature modules of the file manager
ej.filemanager.FileManager.Inject(ej.filemanager.DetailsView,ej.filemanager.
Toolbar,ej.filemanager.NavigationPane);
//Initialize Tab component
var tabObj = new ej.navigations.Tab({
    heightAdjustMode: 'None',
    height: 320,
    showCloseButton: true,
    selecting: onSelect
});
//Render initialized Tab component
tabObj.appendTo('#tab_orientation');
// initialize File Manager component
var fileObject = new ej.filemanager.FileManager({
    ajaxSettings: {
        url: hostUrl + 'api/FileManager/FileOperations',
        getImageUrl: hostUrl + 'api/FileManager/GetImage',
        uploadUrl: hostUrl + 'api/FileManager/Upload',
        downloadUrl: hostUrl + 'api/FileManager/Download'
    },
});
// render initialized FileManager
fileObject.appendTo('#filemanager');
function onSelect() {
    fileObject.refreshLayout();
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 File Manager</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Essential JS 2 File Manager
Component">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
filemanager/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="tab_orientation">
        <div class="e-tab-header">
            <div>Overview</div>
            <div>File Manager</div>
        </div>
        <div class="e-content">
            <div>
                <div class="content-title">
                    <div class="cnt-text">Overview</div>
                </div>
                <div style="font-size:14px">The file manager component
contains a context menu for performing file operations, large-icons view for
displaying the files and folders, and a breadcrumb for navigation. However,
these basic functionalities can be extended by using the additional feature
modules like toolbar, navigation pane, and details view to simplify the
navigation and file operations within the file system.</div>
            </div>
            <div>
                <div class="content-title">
                    <div class="cnt-text">File manager with default
functionalities</div>
                </div>
                <div id="filemanager"></div>
            </div>
        </div>
    </div>
</body>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Change the localization content in ##Platform_Name## File manager control

The example below shows how to modify the file manager component localization content. The upload text of the file manager component can be changed in the following example.

INDEX.JS

```
//Defining texts and messages corresponding to German culture
ej.base.L10n.load({
  'en': {
    'filemanager': {
      // Change the File Upload text.
      "File-Upload": "Files to Upload",
      // Change the Empty folder text.
      "Folder-Empty": "Empty Folder",
    }
  }
})
var hostUrl = 'https://ej2-aspcore-service.azurewebsites.net/';
// inject feature modules of the file manager
ej.filemanager.FileManager.Inject(ej.filemanager.DetailsView,ej.filemanager.
Toolbar,ej.filemanager.NavigationPane);
// initialize File Manager component
var filemanagerInstance = new ej.filemanager.FileManager({
  ajaxSettings: {
    url: hostUrl + 'api/FileManager/FileOperations',
    getImageUrl: hostUrl + 'api/FileManager/GetImage',
    uploadUrl: hostUrl + 'api/FileManager/Upload',
    downloadUrl: hostUrl + 'api/FileManager/Download'
  },
  //defining the locale for File Manager
  locale: 'en'
});
// render initialized File Manager
filemanagerInstance.appendTo('#filemanager');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 File Manager</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 File Manager
Component">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-layouts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-filemanager/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="filemanager"></div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Create the custom file provider using NodeJS

Here we manipulate the Azure Blob Storage to supply the necessary data for the File Manager. We achieve this by utilizing NodeJS to fetch the required data from the Azure blob storage.

NodeJS acts as the bridge between the File Manager component and Azure Blob Storage, allowing seamless communication and data retrieval. Through this integration, the File Manager can access and interact with the data stored in Azure Blob Storage, enabling smooth file management operations.

Prerequisites

- Valid Azure blob storage account. (accountName, accountKey, endpointSuffix)
- Node version 14 above.

Introduction to Azure Blob Storage

Azure Blob Storage is a cloud-based object storage service provided by Microsoft Azure. It is designed to store and manage unstructured data, also known as "blobs" in the cloud. Blobs can be any type of data, such as images, videos, documents, backups, logs, and more.

Key concepts of Azure Blob Storage

Containers: In Azure Blob Storage, data is organized into containers. Containers are logical units that can hold one or more blobs. Think of them as directories or folders that help organize the data.

Blobs: Blobs are the actual data objects stored in Azure Blob Storage.

By understanding the fundamental concepts and use cases of Azure Blob Storage, you will be well-prepared to proceed with setting up and interacting with it using NodeJS in the custom File Provider.

[Create NodeJS project](#)

Following the steps to create the NodeJS project.

Create a new directory for your project and run the following command to initialize a new NodeJS project. This will create a package.json file.

```
`ts
npm init
`
```

Install the following packages.

- express
- @azure/storage-blob
- archiver
- body-parser
- cors
- esm
- multer

Open your text editor or integrated development environment (IDE) and create the index.js file start writing your NodeJS code. This file will serve as the entry point of your application.

```
`ts
const express = require('express');
const app = express();
const port = 3000;
app.get('/', (req, res) => {
res.send('Hello, NodeJS!');
});
app.listen(port, () => {
console.log(Server running on http://localhost:${port});
});
`
```

To start your NodeJS application, simply run the following command in your terminal, pointing to the entry point file:

```
`ts
node index.js
`
```

Initialize container client

We need to first get the BlobServiceClient. By using the connection string, we can obtain the BlobServiceClient. So, format the connection string as shown below.

```
`ts
Const connectionString =
DefaultEndpointsProtocol=https;AccountName=${accountName};AccountKey=${accountKey};E
ndpointSuffix=${EndpointSuffix};
`
```

We can obtain the BlobServiceClient and the **containerClient** using this connection String and the BlobServiceClient. the **containerName** is the container from your Azure blob storage account that you need to access.

```
`ts
import { BlobServiceClient } from "@azure/storage-blob";
const blobServiceClient = BlobServiceClient.fromConnectionString(connectionString);
const containerClient = blobServiceClient.getContainerClient(containerName);
`
```

File actions

Need to provide the following action to creating a new folder, copying and moving of files or folders, deleting, uploading, and downloading the files or folders in the file system

Read

Specify the directory name that needs to be accessed.

```
`ts
const directoryName = 'Files';
`
```

Create the **app.post** method with URL **‘/fileManager’**.

To identify the action by use this condition **req.body.action === ‘read’**

The following table represents the request parameters of **read** operations.

Parameter	Type	Default	Explanation
----	----	----	----
action	String	read	Name of the file operation.
path	String	-	Relative path from which the data has to be read.
showHiddenItems	Boolean	-	Defines show or hide the hidden items.
data	FileManagerDirectoryContent	-	Details about the current path (directory).

Example for request:

```
`ts
{
```

```
action: "read",
path: "/Videos/",
showHiddenItems: false,
data: [
0:{
name:"Videos",
size:0,
dateModified:"2023-09-14T14:28:27.000Z",
dateCreated: "2023-09-14T11:16:57.000Z",
hasChild:true,
isFile:false,
type:"Directory",
filterPath:"/",
fmicon: "e-fe-folder",
fmiconClass: "e-fe-folder",
fmid: "fetree0",
fmmodified: "September 14, 2023 19:58"
}
]
}
```

The following table represents the response parameters of **read** operations.

Parameter	Type	Default	Explanation
----	----	----	----
cwd	FileManagerDirectoryContent	-	Path (Current Working Directory) details.
files	FileManagerDirectoryContent[]	-	Details of files and folders present in given path or directory.
error	ErrorDetails	-	Error Details

The following table represents the contents of **FileManagerDirectoryContent** in the file manager request and response.

Parameter	Type	Default	Explanation	Is required
----	----	----	----	----
name	String	-	File name	Yes

dateCreated	String	-	Date in which file was created (UTC Date string).	Yes
dateModified	String	-	Date in which file was last modified (UTC Date string).	Yes
filterPath	String	-	Relative path to the file or folder.	Yes
hasChild	Boolean	-	Defines this folder has any child folder or not.	Yes
isFile	Boolean	-	Say whether the item is file or folder.	Yes
size	Number	-	File size	Yes
type	String	-	File extension	Yes
permission	[AccessRules](#)	-	File extension	Optional
caseSensitive	Boolean	-	Defines search is case sensitive or not.	Optional
action	String	read	Name of the file operation.	Optional
names	String[]	-	Name list of the items to be downloaded.	Optional
data	FileManagerDirectoryContent	-	Details of the download item.	Optional
uploadFiles	IList<IFormFile>	-	File that are uploaded.	Optional
newName	String	-	New name for the item.	Optional
searchString	String	-	String to be searched in the directory.	Optional
targetPath	String	-	Relative path where the items to be pasted are located.	Optional
targetData	FileManagerDirectoryContent	-	Details of the copied item.	Optional
renameFiles	String[]	-	Details of the renamed item.	Optional

The following table represents the **AccessRules** properties available for file and folder:

Properties	Applicable for file	Applicable for folder	Description
---	---	---	---
Copy	Yes	Yes	Allows access to copy a file or folder.
Read	Yes	Yes	Allows access to read a file or folder.
Write	Yes	Yes	Allows permission to write a file or folder.
WriteContents	No	Yes	Allows permission to write the content of folder.
Download	Yes	Yes	Allows permission to download a file or folder.
Upload	No	Yes	Allows permission to upload to the folder.
Path	Yes	Yes	Specifies the path to apply the rules, which are defined.
Role	Yes	Yes	Specifies the role to which the rule is applied.
IsFile	Yes	Yes	Specifies whether the rule is specified for folder or file.

Example for response:

`ts

```
{
  cwd:
  {
    filterPath: "/",
    hasChild: true,
    name: "Videos",
    size: 0,
    type: "File Folder"
  },
  files:[
    0:{
      dateCreated: "2023-09-14T11:16:57.000Z"
      dateModified: "2023-09-14T11:16:57.000Z"
      filterPath: "/Videos/"
      hasChild: false
      isFile: true
      name: "about.txt"
      size: 29
      type: ".txt"
    }
  ],
  error:null
},
```

[Get image](#)

Create the **app.get** method with URL **‘/fileManager/GetImage’**.

The following table represents the request parameters of **GetImage** operations.

Parameter	Type	Default	Explanation
-----------	------	---------	-------------

----	----	----	----
------	------	------	------

path	String	-	Relative path to the image file
------	--------	---	---------------------------------

The req.query.path contains the exact path of the images. For example: **"/Jack.png"**.

Download the blob (image) from Azure Blob Storage using the blobClient and stores the result in the downloadResponse variable.

Pipe the `readableStreamBody` from the blob to the `res` response. It means the image data will be streamed from the Azure Blob Storage directly to the client's browser when the image URL is accessed.

Handle the exception if the image is not available in the given path.

Download

Create the **`app.post`** method with URL **`'/fileManager/Download'`**.

The following table represents the request parameters of *download* operations.

Parameter	Type	Default	Explanation
----	----	----	----
action	String	download	Name of the file operation
path	String	-	Relative path to location where the files to download are present.
names	String[]	-	Name list of the items to be downloaded.
data	FileManagerDirectoryContent	-	Details of the download item.

Example for request:

```
`ts
{
  action: 'download',
  path: '/Downloads/Testing/',
  names: [ 'About.txt' ],
  data: [
    0:{
      name: 'About.txt',
      type: '.txt',
      isFile: true,
      size: 29,
      dateModified: '2023-09-14T06:03:52.000Z',
      hasChild: false,
      filterPath: '/Downloads/Testing/',
      fmcreated: null,
      fmmodified: 'September 14, 2023 11:33',
      fmiconClass: 'e-fe-txt',
      fmicon: 'e-fe-txt'
    }
  ]
}
```

The **req.body.downloadInput** must be parsed to get the **downloadObj**. Download the blob from Azure Blob Storage using the blobClient.

Download the blob from Azure Blob Storage using the blobClient and Pipe the readableStreamBody to the response object.

Create the archive file to download the multiple Files, Folders and single folders, then pipe the archive to the response.

Upload

Create the **app.post** method with URL **'/fileManager/Upload'**.

The following table represents the request parameters of *Upload* operations.

Parameter	Type	Default	Explanation
action	String	Save	Name of the file operation.
path	String	-	Relative path to the location where the file has to be uploaded.
uploadFiles	IList<IFormFile>	-	File that are uploaded.

Example for request:

```
`ts
{
  path: '/Pictures/',
  action: 'save',
  data: [
    0:{
      name: 'Pictures',
      type: 'File Folder',
      isFile: true,
      size: 0,
      dateModified: '2023-09-14T06:03:52.000Z',
      hasChild: true,
      filterPath: '',
      fmid: 'fetree1',
    }
  ],
  filename: 'bird (2).jpg'
}
```

Multer is a popular middleware used to handle file uploads in Express-based web applications. Create the Multer config to store the upload files in buffer.

```
`ts
const multerConfig = {
  storage: memoryStorage()
};
`
```

Need to handle the 3 cases here.

- Save
- Keep Both (action name will be **keepboth**)
- Replace (action name will be **replace**)

Create the **getBlockBlobClient** with the **req.body.filename**. If the blob does not exist, then upload the data to that blob. If the blob already exists, then create an error message containing "File Already Exists" and send the response.

Create a new folder

The following table represents the request parameters of *create* operations.

Parameter	Type	Default	Explanation
----	----	----	----
action	String	create	Name of the file operation.
path	String	-	Relative path in which the folder has to be created.
name	String	-	Name of the folder to be created.
data	FileManagerDirectoryContent	-	Details about the current path (directory).

Example for request:

```
`ts
action: "create",
data: [
  0:{
    filterPath: "/",
    hasChild: true,
    isFile: false,
    name: "files",
    nodeId: "fe_tree",
    size: 0,
```

```
type: ""
}
],
name: "Hello",
path: "/test/"
,
```

Check the existence of the folder, If the folder exists then send the error message containing “Folder already exists”. If it does not exist, then create the folder. Create the folder by creating the file in that folder’s path.

The following table represents the response parameters of *create* operations.

Parameter	Type	Default	Explanation
files	FileManagerDirectoryContent[]	-	Details of the created folder
error	ErrorDetails	-	Error Details

Example for response:

```
`ts
{
  cwd: null,
  files: [
    0:{
      dateCreated: "2023-09-14T10:52:25.000Z",
      dateModified: "2023-09-14T10:52:25.000Z",
      filterPath: null,
      hasChild: false,
      isFile: false,
      name: "New",
      size: 0,
      type: "Directory"
    }
  ],
  details: null,
  error: null
},
```

Rename

The following table represents the request parameters of *rename* operations.

Parameter	Type	Default	Explanation
	----	----	----
action	String	rename	Name of the file operation.
path	String	-	Relative path in which the item is located.
name	String	-	Current name of the item to be renamed.
newName	String	-	New name for the item.
data	FileManagerDirectoryContent	-	Details of the item to be renamed.

Example for request:

```
`ts
{
  action: "rename",
  data: [
    0:{
      dateCreated: "2023-09-14T10:41:17.000Z",
      filterPath: "/Pictures/Nature/",
      hasChild: false,
      iconClass: "e-fe-image",
      isFile: true,
      name: "seaviews.jpg",
      size: 95866,
      type: ".jpg"
    }
  ],
  newName: "seaview.jpg",
  name: "seaviews.jpg",
  path: "/Pictures/Nature/"
}
```

Renaming can be done by copy the folder or file from the source blob instance to target blob instance. If the file exists, then send the error message as response.

The following table represents the response parameters of *rename* operations.

Parameter	Type	Default	Explanation
-----------	------	---------	-------------

```
|----|----|----|----|
|files|FileManagerDirectoryContent[]|-|Details of the renamed item.|
|error|ErrorDetails|-|Error Details|
```

Example for response:

```
`ts
{
  cwd:null,
  files:[
    0:{
      name:"seaview.jpg",
      size:95866,
      dateModified:"2023-09-14T11:16:57.000Z",
      dateCreated:"2023-09-14T10:41:17.000Z",
      hasChild:false,
      isFile:true,
      type:".jpg",
      filterPath:"/Pictures/Nature/"
    }
  ],
  error:null,
  details:null
}
```

Delete

The following table represents the request parameters of *delete* operations.

Parameter	Type	Default	Explanation
action	String	delete	Name of the file operation.
path	String	-	Relative path where the items to be deleted are located.
names	String[]	-	List of the items to be deleted.
data	FileManagerDirectoryContent	-	Details of the item to be deleted.

Example for request:

```
`ts
```



```
{
  action: "delete",
  path: "/",
  names: ["bird.jpg"],
  data: [
    0:{
      dateModified: "2023-09-14T09:12:53.000Z",
      filterPath: "/",
      hasChild: false,
      iconClass: "e-fe-image",
      isFile: true,
      name: "bird.jpg",
      size: 102182,
      type: ".jpg"
    }
  ]
}
```

To delete the file, directly get the file instance and delete the file. To delete the folder, we need to get all files inside that folder and delete all those files.

Handle the null exception if file or folder is not available.

The following table represents the response parameters of *delete* operations.

Parameter	Type	Default	Explanation
files	FileManagerDirectoryContent[]	-	Details about the deleted item(s).
error	ErrorDetails	-	Error Details

Example for response:

```
`ts
{
  cwd: null,
  details: null,
  error: null,
  files: [
```

```
0:{
  dateModified: "2023-09-14T09:12:53.000Z",
  filterPath: "/",
  hasChild: false,
  iconClass: "e-fe-image",
  isFile: true,
  name: "bird.jpg",
  size: 102182,
  type: ".jpg"
}
]
```

Details

The following table represents the request parameters of *details* operations.

Parameter	Type	Default	Explanation
action	String	details	Name of the file operation.
path	String	-	Relative path where the items are located.
names	String[]	-	List of the items to get details.
data	FileManagerDirectoryContent	-	Details of the selected item.

Example:

```
`ts
{
  action: "details",
  path: "/FileContents/",
  names: ["bird.jpg"],
  data: [
    0:{
      dateModified: "2023-09-14T09:12:53.000Z",
      filterPath: "/",
      hasChild: false,
      iconClass: "e-fe-image",
```

```
isFile: true,  
name: "bird.jpg",  
size: 102182,  
type: ".jpg"  
}  
]  
}  
,
```

To get the file and folder details, iterate the **req.body.names** to get the details of files and folders. If the data is file, then get the file instance and get the properties using the **getProperties** method. If the data is Folder, then get the blobs details under that folder using **listBlobsFlat** method. Get the required properties and send final response. Handled the null exception if the file or folder is not available.

The following table represents the response parameters of *details* operations.

Parameter	Type	Default	Explanation
----	----	----	----
details	FileManagerDirectoryContent	-	Details of the requested item(s).
error	ErrorDetails	-	Error Details

Example:

```
`ts  
{  
  cwd:null,  
  files:null,  
  error:null,  
  details:  
  {  
    created: "2023-09-15T06:04:12.000Z"  
    isFile: true  
    location: "Files/bird.jpg"  
    modified: "2023-09-15T06:04:12.000Z"  
    multipleFiles: false  
    name: "bird.jpg"  
    size: "100.0 KB"  
  }  
}
```

Search

The following table represents the request parameters of *search* operations.

Parameter	Type	Default	Explanation
action	String	search	Name of the file operation.
path	String	-	Relative path to the directory where the files should be searched.
showHiddenItems	Boolean	-	Defines show or hide the hidden items.
caseSensitive	Boolean	-	Defines search is case sensitive or not.
searchString	String	-	String to be searched in the directory.
data	FileManagerDirectoryContent	-	Details of the searched item.

Example for request:

```
`ts
{
  action: "search",
  path: "/asia/",
  searchString: "nature",
  showHiddenItems: false,
  caseSensitive: false,
  data: [
    0:{
      filterPath: "/",
      hasChild: true,
      name: "asia",
      size: 0,
      type: "File Folder",
      fmid: "fetree1"
    }
  ]
}
```

Replace the '*' in the **req.body.searchString** and assign the result to new variable. Get all blobs under this directory and check that path contains the search string

The following table represents the response parameters of *search* operations.

|Parameter|Type|Default|Explanation|

|----|----|----|----|

|cwd|[FileManagerDirectoryContent](#)|-|Path (Current Working Directory) details.|

|files|FileManagerDirectoryContent[]|-|Files and folders in the searched directory that matches the search input.|

|error|[ErrorDetails](#)|-|Error Details|

Example for response:

```
`ts
{
  cwd:
  {
    name:"asia",
    size:0,
    dateModified:"2023-09-14T14:28:27.000Z",
    dateCreated:"2023-09-14T11:16:57.000Z",
    hasChild:true,
    isFile:false,
    type:"File Folder",
    filterPath:"/"
  },
  files:[
    0: {
      dateModified: "2023-09-15T06:22:00.000Z",
      filterPath: "/asia/",
      hasChild: false,
      isFile: true,
      name: "about.txt",
      size: 42,
      type: ".txt"
    }
  ],
  error:null,
  details:null
```

```
}  
、
```

Copy and move

The following table represents the request parameters of *copy* operations.

Parameter	Type	Default	Explanation
----	----	----	----
action	String	copy	Name of the file operation.
path	String	-	Relative path to the directory where the files should be copied.
names	String[]	-	List of files to be copied.
targetPath	String	-	Relative path where the items to be pasted are located.
data	FileManagerDirectoryContent	-	Details of the copied item.
targetData	FileManagerDirectoryContent	-	Details of the copied item.
renameFiles	String[]	-	Details of the renamed item.

Example for request:

```
`ts  
{  
  action: "copy",  
  path: "/",  
  names: ["bird.jpg"],  
  renameFiles: [],  
  targetPath: "/asia/",  
  targetData: {  
    filterPath: "/",  
    hasChild: true,  
    name: "asia",  
    size: 0,  
    type: "File Folder",  
    fmid: "fetree1",  
  },  
  data: [  
    0:{  
      dateCreated: "2023-09-15T06:04:12.000Z",  
      dateModified: "2023-09-15T06:04:12.000Z",
```

```
filterPath: "/",
hasChild: false,
isFile: true,
name: "bird.jpg",
size: 102182,
type: ".jpg",
fmcreated: "September 15, 2023 11:34",
fmhtmlAttr: {class: "e-large-icon", title: "bird.jpg"},
fmiconClass: "e-fe-image",
fmimageAttr: {alt: "bird.jpg"},
fmimageUrl: "http://localhost:3000/GetImage?path=%2Fbird.jpg&time=1694760243307",
fmmodified: "September 15, 2023 11:34",
}
]
}
`
```

Action name will be **move** for move action.

The following table represents the response parameters of *copy* operations.

Parameter	Type	Default	Explanation
----	----	----	----
cwd	FileManagerDirectoryContent	-	Path (Current Working Directory) details.
files	FileManagerDirectoryContent[]	-	Details of copied files or folders
error	ErrorDetails	-	Error Details

Example for response:

```
`ts
{
  cwd:null,
  files:[
    0:{
      dateCreated: "2023-09-15T06:55:03.000Z"
      dateModified: "2023-09-15T06:55:03.000Z"
      filterPath: "/asia/"
      hasChild: false
```

```

isFile: true
name: "bird.jpg"
previousName: null
size: 102182
type: ".jpg"
}
],
error:null,
details:null
}
,

```

Need to handle two cases.

- Directory copy and move.
- File copy and move.

Create the **isRename** variable to store the is request is rename or not. If the **isRename** is false then check the existence of the folders, and if folder is existing, then send the error message. If **isRename** is true, then don't check the existence of the folder.

To move or copy the folders you need to get all the blobs from that folder and create the new path for each blob and copy the data from the old path to the new path. To move or copy the files copy the data from the source blob client to target client. If the action is move then delete the old blob.

Note: To get the complete project, refer to this [link](#)

Floating Action Button

Form validator

Validation rules in `##Platform_Name##` Form validator control

Default Rules

The **FormValidator** has following default validation rules, which are used to validate the form.

Rules	Description	Example
-----	-----	-----
required	The form input element must have any input values	a or 1 or -
email	The form input element must have valid email format values	<form@syncfusion.com>
url	The form input element must have valid url format values	http://syncfusion.com/
date	The form input element must have valid date format values	05/15/2017
dateISO	The form input element must have valid dateISO format values	2017-05-15

| **number** | The form input element must have valid **number** format values | 1.0 or 1 |

| **digits** | The form input element must have valid **digits** values | 1 |

| **maxLength** | Input value must have less than or equal to **maxLength** character length | if
maxLength: 5, **check** is valid and **checking** is invalid |

| **minLength** | Input value must have greater than or equal to **minLength** character length | if
minLength: 5, **testing** is valid and **test** is invalid |

| **rangeLength** | Input value must have value between **rangeLength** character length | if
rangeLength: [4,5], **test** is valid and **key** is invalid |

| **range** | Input value must have value between **range** number | if **range**: [4,5], **4** is valid and **6** is
invalid |

| **max** | Input value must have less than or equal to **max** number | if **max**: 3, **3** is valid and **4** is invalid |

| **min** | Input value must have greater than or equal to **min** number | if **min**: 4, **5** is valid and **2** is invalid
|

Error messages in ##Platform_Name## Form validator control

The **FormValidator** provides default error messages for all default validation rules.

It is tabulated as follows

Rules	message
-----	-----
required	This field is required.
email	Please enter a valid email address.
url	Please enter a valid URL.
date	Please enter a valid date.
dateIso	Please enter a valid date (ISO).
number	Please enter a valid number.
digit	Please enter only digits.
maxLength	Please enter no more than {0} characters.
minLength	Please enter at least {0} characters.
rangeLength	Please enter a value between {0} and {1} characters long.
range	Please enter a value between {0} and {1}.
max	Please enter a value less than or equal to {0}.
min	Please enter a value greater than or equal to {0}.
regex	Please enter a correct value.

Customizing Error Messages

The default error message for a rule can be customizable by defining it along with concern rule object as follows.

INDEX.JS

```
var customFn = (args) => {
    return args['value'].length >= 5;
};
var options = {
    rules: {
        'user': { required: true },
        'password': { minLength: [customFn, 'Need atleast 5 letters'] }
    }
};
var formObject = new ej.inputs.FormValidator('#form-element', options);
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Form Validator</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <form id="form-element" class="form-horizontal">
      <div class="form-group">
        <label class="col-sm-3 control-label" for="user">User
Name</label>
        <div class="col-sm-6">
          <input type="text" id="required" name="user"
class="form-control">
        </div>
      </div>
      <div class="form-group">
        <label class="col-sm-3 control-label"
for="password">Password</label>
        <div class="col-sm-6">
          <input type="password" id="number" name="password"
class="form-control">
```

```

        </div>
    </div>
</form>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

INDEX.CSS

```

#container {
    visibility: hidden;
    padding: 10px;
}
#loader {
    color: #008cff;
    height: 40px;
    width: 30%;
    position: absolute;
    top: 45%;
    left: 45%;
}

```

Customizing Error Placement

The `FormValidator` has an event [customPlacement](#) which can be used to place the error message from default position to desired custom location.

INDEX.JS

```

var options = {
    rules: {
        'user': { required: [true, 'User Name: required'] },
        'password': { minLength: [5, 'Password: need atleast 5 characters'] }
    },
    customPlacement: (inputElement, error) => {
        document.getElementById('error').appendChild(error);
    }
};
var formObject = new ej.inputs.FormValidator('#form-element', options);

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Form Validator</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <form id="form-element" class="form-horizontal">
            <div class="form-group">
                <label class="col-sm-3 control-label" for="user">User
Name</label>
                <div class="col-sm-6">
                    <input type="text" id="required" name="user"
class="form-control">
                </div>
            </div>
            <div class="form-group">
                <label class="col-sm-3 control-label"
for="password">Password</label>
                <div class="col-sm-6">
                    <input type="password" id="number" name="password"
class="form-control">
                </div>
            </div>
            <div class="form-group">
                <div class="col-sm-3 control-label"></div>
                <div class="col-sm-6">
                    <div id="error"></div>
                </div>
            </div>
        </form>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

INDEX.CSS

```

#container {
    visibility: hidden;
    padding: 10px;
}
#loader {

```

```

    color: #008cff;
    height: 40px;
    width: 30%;
    position: absolute;
    top: 45%;
    left: 45%;
}
#error {
    width: 100%;
    height: 70px;
    border: 1px solid red;
    padding: 10px;
}

```

Localization in ##Platform_Name## Form validator control

The [Localization](#) library allows users to localize the default error message contents of the `formValidator` to different cultures using the `locale` property.

The `FormValidator` provides default error messages for all default validation rules. It is tabulated as follows:

Rules	message
required	This field is required.
email	Please enter a valid email address.
url	Please enter a valid URL.
date	Please enter a valid date.
dateIso	Please enter a valid date (ISO).
number	Please enter a valid number.
digit	Please enter only digits.
maxLength	Please enter no more than {0} characters.
minLength	Please enter at least {0} characters.
rangeLength	Please enter a value between {0} and {1} characters long.
range	Please enter a value between {0} and {1}.
max	Please enter a value less than or equal to {0}.
min	Please enter a value greater than or equal to {0}.
regex	Please enter a correct value.

Loading translations

To load translation object in your application use `load` function of `L10n` class.

The following example demonstrates the `FormValidator` in `Arabic` culture with error message for email.

INDEX.JS

```

var L10n = ej.base.L10n;
L10n.load({
  'ar-AR': {
    'formValidator': {
      "required": "هذه الخانة مطلوبه",
      "email": "أدخل بريد إلكتروني صالح",
      "url": "أدخل رابط صحيح من فضلك",
      "date": "أرجوك ادخل تاريخ صحيح",
      "dateIso": "الرجاء إدخال تاريخ صالح (ISO)",
      "creditcard": "يرجى إدخال رقم بطاقة صالح",
      "number": "من فضلك أدخل رقما صالحا",
      "digits": "الرجاء إدخال الأرقام فقط",
      "maxLength": "الرجاء إدخال ما لا يزيد عن {0} حرف",
      "minLength": "الرجاء إدخال أحرف {0} على الأقل",
      "rangeLength": "يرجى إدخال قيمة بين {0} و {1} من الأحرف",
      "range": "يرجى إدخال قيمة بين {0} و {1}",
      "max": "الرجاء إدخال قيمة أقل من أو تساوي {0}",
      "min": "الرجاء إدخال قيمة أكبر من أو تساوي {0}",
      "regex": "يرجى إدخال قيمة صحيحة",
      "tel": "يرجى إدخال رقم هاتف صالح",
      "pattern": "الرجاء إدخال قيمة نمط صحيح",
      "equalTo": "يرجى إدخال نص مطابقة صحيح"
    }
  }
});
var options = {
  rules: {
    email: { email: true },
  },
  locale: "ar-AR"
}
var formObject = new ej.inputs.FormValidator('#form-element', options);

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Form Validator</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>

```

```
<body>

  <div id="container">
    <form id="form-element" class="form-horizontal">
      <div class="form-group">
        <label class="col-sm-3 control-label"
for="user">Email</label>
        <div class="col-sm-6">
          <input type="text" id="email" name="email" class="form-
control">
        </div>
      </div>
    </form>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

INDEX.CSS

```
#container {
  visibility: hidden;
  padding: 10px;
}
#loader {
  color: #008cff;
  height: 40px;
  width: 30%;
  position: absolute;
  top: 45%;
  left: 45%;
}
#error {
  width: 100%;
  height: 70px;
  border: 1px solid red;
  padding: 10px;
}
```

Gantt

Module in ##Platform_Name## Gantt control

The modules that are available in Gantt are as follows.

Module	Description
-----	-----
Sort	Inject this module to use sorting feature.

[Filter](#)	Inject this module to use filtering feature.
[Reorder](#)	Inject this module to use reorder feature.
[ExcelExport](#)	Inject this module to use excel export feature.
[PdfExport](#)	Inject this module to use PDF export feature.
[RowDD](#)	Inject this module to use row drag and drop feature.
[Resize](#)	Inject this module to use resize feature.
[Toolbar](#)	Inject this module to use toolbar feature.
[Edit](#)	Inject this module is use editing feature.
[Selection](#)	Inject this module to use selection feature.
[DayMarkers](#)	Inject this module to use event markers.
[ContextMenu](#)	Inject this module to use context menu feature.
[ColumnMenu](#)	Inject this module to use column menu feature.
[VirtualScroll](#)	Inject this module to use virtual scroll feature.
[CriticalPath](#)	Inject this module to use critical path feature.

These modules should be injected into the Gantt using the **Gantt.Inject** method.

Data binding in ##Platform_Name## Gantt control

The Gantt control uses **DataManager** for binding the data source, which supports both RESTful JSON data services and local JavaScript object array. The [dataSource](#) property can be assigned either with the instance of **DataManager** or JavaScript object array collection. The Gantt control supports binding two types of data:

- Local data
- Remote data

Local data

To bind local data to Gantt, you can assign a JavaScript object array to the [dataSource](#) property. The local data source can also be provided as an instance of the **DataManager**.

In local data binding, the data source for rendering the Gantt control is retrieved from the same application locally.

The following are the two types of data binding possible with the Gantt control:

- Hierarchical data binding.
- Self-referential data binding (Flat data).

Hierarchical data binding

The [child](#) property is used to map the child records in hierarchical data.

The following code example shows how to bind the hierarchical local data into the Gantt control.

INDEX.JS


```

var GanttData = [
    {
        TaskID: 1,
        TaskName: 'Project Initiation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        subtasks: [
            { TaskID: 2, TaskName: 'Identify Site location', StartDate:
new Date('04/02/2019'), Duration: 4, Progress: 50 },
            { TaskID: 3, TaskName: 'Perform Soil test', StartDate: new
Date('04/02/2019'), Duration: 4, Progress: 50 },
            { TaskID: 4, TaskName: 'Soil test approval', StartDate: new
Date('04/02/2019'), Duration: 4, Progress: 50 },
        ]
    },
    {
        TaskID: 5,
        TaskName: 'Project Estimation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        subtasks: [
            { TaskID: 6, TaskName: 'Develop floor plan for estimation',
StartDate: new Date('04/04/2019'), Duration: 3, Progress: 50 },
            { TaskID: 7, TaskName: 'List materials', StartDate: new
Date('04/04/2019'), Duration: 3, Progress: 50 },
            { TaskID: 8, TaskName: 'Estimation approval', StartDate: new
Date('04/04/2019'), Duration: 3, Progress: 50 }
        ]
    },
];

var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    }
});
ganttChart.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Self-referential data binding (Flat data)

The Gantt control can be bound with self-referential data by mapping the data source field values to the [id](#) and [parentID](#) properties.

- ID field: This field contains unique values used to identify each individual task and it is mapped to the [id](#) property.
- Parent ID field: This field contains values that indicate parent tasks and it is mapped to the [parentID](#) property.

INDEX.JS

```

var SelfReferenceData = [
    { TaskID: 1, TaskName: 'Project Initiation', StartDate: new
Date('04/02/2019'), EndDate: new Date('04/21/2019') },
    { TaskID: 2, TaskName: 'Identify Site location', StartDate: new
Date('04/02/2019'), Duration: 4, Progress: 50, ParentId: 1 },
    { TaskID: 3, TaskName: 'Perform Soil test', StartDate: new
Date('04/02/2019'), Duration: 4, Progress: 50, ParentId: 1 },
    { TaskID: 4, TaskName: 'Soil test approval', StartDate: new
Date('04/02/2019'), Duration: 4, Progress: 50, ParentId: 1 },
    { TaskID: 5, TaskName: 'Project Estimation', StartDate: new
Date('04/02/2019'), EndDate: new Date('04/21/2019') },
    { TaskID: 6, TaskName: 'Develop floor plan for estimation',
StartDate: new Date('04/04/2019'), Duration: 3, Progress: 50, ParentId: 5 },
    { TaskID: 7, TaskName: 'List materials', StartDate: new
Date('04/04/2019'), Duration: 3, Progress: 50, ParentId: 5 },
    { TaskID: 8, TaskName: 'Estimation approval', StartDate: new
Date('04/04/2019'), Duration: 3, Progress: 50, ParentId: 5 }
];
var ganttChart = new ej.gantt.Gantt({
    dataSource: SelfReferenceData,
    height: '450px',

```

```

        taskFields: {
            id: 'TaskID',
            name: 'TaskName',
            startDate: 'StartDate',
            duration: 'Duration',
            progress: 'Progress',
            parentID: 'ParentId'
        }
    });
    ganttChart.appendTo('#Gantt');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Remote data

To bind remote data to the Gantt component, assign service data as an instance of **DataManager** to the [dataSource](#) property.

INDEX.JS

```

var hostUrl = 'https://ej2services.syncfusion.com/production/web-services/';
var GanttData = new ej.data.DataManager({
    url: hostUrl + 'api/GanttData',
    adaptor: new ej.data.WebApiAdaptor(),
    crossDomain: true
});
```

```

});
var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskId',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        dependency: 'Predecessor',
        child: 'SubTasks'
    }
});
ganttChart.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

URL Adaptor

Please refer the [link](#) to create the ADO.NET Entity Data Model in Visual studio,

We can define data source for Gantt as instance of DataManager using url property of DataManager. Please Check the below code snippet to assign data source to Gantt.

```
`ts
```

```

import { Gantt } from '@syncfusion/ej2-gantt';
import { DataManager, UrlAdaptor } from '@syncfusion/ej2-data';
let dataSource: DataManager = new DataManager({
url: '/Home/UrlDatasource',
adaptor: new UrlAdaptor
});
let gantt: Gantt = new Gantt({
dataSource: dataSource,
height: '450px',
treeColumnIndex: 1,
taskFields: {
id: 'TaskId',
name: 'TaskName',
startDate: 'StartDate',
progress: 'Progress',
duration: 'Duration',
dependency: 'Predecessor',
child: 'SubTasks'
}
});
gantt.appendTo('#Gantt');
`ts
GanttDataSourceEntities db = new GanttDataSourceEntities();
public ActionResult UrlDatasource(DataManagerRequest dm)
{
List<GanttData>DataList = db.GanttDatas.ToList();
var count = DataList.Count();
return Json(new { result = DataList, count = count });
}
`

```

Load child on demand

To render child records on demand, assign a remote service URL in the instance of DataManager to the `url` property. To interact with the remote data source, provide the endpoint URL and also define the [hasChildMapping](#) property in taskFields of Gantt Chart.

The `hasChildMapping` property maps the field name in the data source, which denotes whether the current record holds any child records. This is useful internally to show expand icon while binding child data on demand.

When [loadChildOnDemand](#) is disabled, all the root nodes are rendered in a collapsed state at initial load. On expanding the root node, the child nodes will be loaded from the remote server.

When `enableVirtualization` is enabled and `loadChildOnDemand` is disabled, only the current viewport root nodes are rendered in a collapsed state.

When a root node is expanded, its child nodes are rendered and maintained in a collection locally, such that on consecutive expand/collapse actions on the root node, the child nodes are loaded locally instead of from the remote server.

When the `loadChildOnDemand` is enabled, parent records are rendered in an expanded state.

`ts

```
import { Gantt, VirtualScroll, Selection } from '@syncfusion/ej2-gantt';
import { DataManager, WebApiAdaptor } from '@syncfusion/ej2-data';
import './App.css';

Gantt.Inject(Selection, VirtualScroll);

let dataSource: DataManager = new DataManager({
  url: 'https://ej2services.syncfusion.com/js/development/api/GanttLoadOnDemand',
  adaptor: new WebApiAdaptor,
  crossDomain: true
});

let gantt: Gantt = new Gantt({
  dataSource: dataSource,
  loadChildOnDemand: false,
  taskFields: {
    id: 'taskId',
    name: 'taskName',
    startDate: 'startDate',
    endDate: 'endDate',
    duration: 'duration',
    progress: 'progress',
```

```
hasChildMapping: 'isParent',
parentID: 'parentID'
},
columns: [
{ field: 'taskId', headerText: 'Task ID' },
{ field: 'taskName', headerText: 'Task Name', allowReordering: false },
{ field: 'startDate', headerText: 'Start Date', allowSorting: false },
{ field: 'duration', headerText: 'Duration', allowEditing: false },
{ field: 'progress', headerText: 'Progress', allowFiltering: false },
],
allowSelection: true,
enableVirtualization: true,
splitterSettings: {
columnIndex: 3,
},
tooltipSettings: {
showTooltip: true
},
highlightWeekends: true,
timelineSettings: {
showTooltip: true,
topTier: {
unit: 'Week',
format: 'dd/MM/yyyy'
},
bottomTier: {
unit: 'Day',
count: 1
}
},
treeColumnIndex: 1,
taskbarHeight: 20,
rowHeight: 40,
```

```

height: '460px',
projectStartDate: new Date('01/02/2000'),
projectEndDate: new Date('12/01/2002'),
});
ganttt.appendTo('#Gantt');
`

```

The following code example describes handling of Load on demand at server end.

```

`ts
public object Get()
{
    DataOperations operation = new DataOperations();
    var queryString = Request.Query;
    if (tree.Count == 0)
        tree = TreeData.GetTree();
    if (queryString.Keys.Contains("$filter") && !queryString.Keys.Contains("$stop"))
    {
        StringValues filter;
        queryString.TryGetValue("$filter", out filter);
        int? fltr;
        if (filter[0].ToString().Split("eq")[1] == " null")
        {
            fltr = null;
        }
        else
        {
            fltr = Int32.Parse(filter[0].ToString().Split("eq")[1]);
        }
        IQueryable<TreeData> data1 = tree.Where(f => f.parentID == fltr).AsQueryable();
        return new { result = data1.ToList(), count = data1.Count() };
    }
    StringValues expand;
    queryString.TryGetValue("$expand", out expand);

```



```
if (queryString.Keys.Contains("$expand")) // setting the ExpandStateMapping property whether is true
or false
{
if (expand[0].ToString().Split(",")[0] == "ExpandingAction")
{
var val = TreeData.GetTree().Where(ds => ds.taskId ==
int.Parse(expand[0].ToString().Split(",")[1])).FirstOrDefault();
val.IsExpanded = true;
}
else if (expand[0].ToString().Split(",")[0] == "CollapsingAction")
{
var val = TreeData.GetTree().Where(ds => ds.taskId ==
int.Parse(expand[0].ToString().Split(",")[1])).FirstOrDefault();
val.IsExpanded = false;
}
}
List<TreeData> data = tree.ToList();
if (queryString.Keys.Contains("$select"))
{
data = (from ord in tree
select new TreeData
{
parentID = ord.parentID
}
).ToList();
return data;
}
data = data.Where(p => p.parentID == null).ToList();
int count = data.Count;
if (queryString.Keys.Contains("$inlinecount"))
{
StringValues Skip;
StringValues Take;
StringValues loadchild;
```

```
int skip = (queryString.TryGetValue("$skip", out Skip)) ? Convert.ToInt32(Skip[0]) : 0;
int top = (queryString.TryGetValue("$top", out Take)) ? Convert.ToInt32(Take[0]) : data.Count();
var GroupData = TreeData.GetTree().ToList().GroupBy(rec => rec.parentID)
.Where(g => g.Key != null).ToDictionary(g => g.Key?.ToString(), g => g.ToList());
foreach (var Record in data.ToList())
{
    if (GroupData.ContainsKey(Record.taskId.ToString()))
    {
        var ChildGroup = GroupData[Record.taskId.ToString()];
        if (ChildGroup?.Count > 0)
            AppendChildren(ChildGroup, Record, GroupData, data);
    }
}
if (expand.Count > 0 && expand[0].ToString().Split(",")[0] == "CollapsingAction")
{
    string IdMapping = "taskId";
    List<WhereFilter> CollapseFilter = new List<WhereFilter>();
    CollapseFilter.Add(new WhereFilter() { Field = IdMapping, value = expand[0].ToString().Split(",")[1],
    Operator = "equal" });
    var CollapsedParentRecord = operation.PerformFiltering(data, CollapseFilter, "and");
    var index = data.Cast<object>().ToList().IndexOf(CollapsedParentRecord.Cast<object>().ToList()[0]);
    skip = index;
}
else if (expand.Count > 0 && expand[0].ToString().Split(",")[0] == "ExpandingAction")
{
    string IdMapping = "taskId";
    List<WhereFilter> ExpandFilter = new List<WhereFilter>();
    ExpandFilter.Add(new WhereFilter() { Field = IdMapping, value = expand[0].ToString().Split(",")[1],
    Operator = "equal" });
    var ExpandedParentRecord = operation.PerformFiltering(data, ExpandFilter, "and");
    var index = data.Cast<object>().ToList().IndexOf(ExpandedParentRecord.Cast<object>().ToList()[0]);
    skip = index;
}
return new { result = data.Skip(skip).Take(top), count = data.Count };
```

```
}  
else  
{  
    return TreeData.GetTree();  
}  
}  
  
private void AppendChildren(List<TreeData> ChildRecords, TreeData ParentItem, Dictionary<string,  
List<TreeData>> GroupData, List<TreeData> data)  
{  
    var queryString = Request.Query;  
    string TaskId = ParentItem.taskId.ToString();  
    var index = data.IndexOf(ParentItem);  
    foreach (var Child in ChildRecords)  
    {  
        string ParentId = Child.parentID.ToString();  
        if (TaskId == ParentId && (bool)ParentItem.IsExpanded)  
        {  
            if (data.IndexOf(Child) == -1)  
                ((IList)data).Insert(++index, Child);  
            if (GroupData.ContainsKey(Child.taskId.ToString()))  
            {  
                var DeepChildRecords = GroupData[Child.taskId.ToString()];  
                if (DeepChildRecords?.Count > 0)  
                    AppendChildren(DeepChildRecords, Child, GroupData, data);  
            }  
        }  
    }  
}  
  
// GET: api/Orders/  
[HttpGet("{id}", Name = "Get")]  
public string Get(int id)  
{  
    return "value";  
}
```

```
}  
[HttpPost]  
public object Post([FromBody] TreeData[] value)  
{  
    //handle insert action  
    for (var i = 0; i < value.Length; i++)  
    {  
        tree.Insert(0, value[i]);  
    }  
    return value;  
}  
//// PUT: api/Orders  
[HttpPut]  
public object Put([FromBody] TreeData[] value)  
{  
    //handle edit action  
    if (value.Length == 1 && value[0].isParent == true)  
    {  
        UpdateDependentRecords(value[0]);  
    }  
    for (var i = 0; i < value.Length; i++) {  
        var ord = value[i];  
        TreeData val = tree.Where(or => or.taskId == ord.taskId).FirstOrDefault();  
        val.taskId = ord.taskId;  
        val.taskName = ord.taskName;  
        val.endDate = ord.endDate;  
        val.startDate = ord.startDate;  
        val.duration = ord.duration;  
        val.predecessor = ord.predecessor;  
    }  
    return value;  
}  
private void UpdateDependentRecords(TreeData ParentItem)
```

```

{
var data = tree.Where(p => p.parentID == ParentItem.taskId).ToList();
var previousData = tree.Where(p => p.taskId == ParentItem.taskId).ToList();
var previousStartDate = previousData[0].startDate;
var previousEndDate = previousData[0].endDate;
double sdiff = (double)GetTimeDifference((DateTime)previousStartDate,
(DateTime)ParentItem.startDate);
double ediff = (double)GetTimeDifference((DateTime)previousEndDate,
(DateTime)ParentItem.endDate);
GetRootChildRecords(ParentItem);
for(var i=0; i<ChildRecords.Count;i++)
{
ChildRecords[i].startDate = ((DateTime)ChildRecords[i].startDate).AddSeconds(sdiff);
ChildRecords[i].endDate = ((DateTime)ChildRecords[i].endDate).AddSeconds(ediff);
}
}
private void GetRootChildRecords(TreeData ParentItem)
{
var currentchildRecords = tree.Where(p => p.parentID == ParentItem.taskId).ToList();
for (var i = 0; i < currentchildRecords.Count; i++) {
var currentRecord = currentchildRecords[i];
ChildRecords.Add(currentRecord);
if (currentRecord.isParent == true)
{
GetRootChildRecords(currentRecord);
}
}
}
public object GetTimeDifference(DateTime sdate, DateTime edate)
{
return new DateTime(edate.Year, edate.Month, edate.Day, edate.Hour, edate.Minute, edate.Second,
DateTimeKind.Utc).Subtract(new DateTime(sdate.Year, sdate.Month, sdate.Day, sdate.Hour,
sdate.Minute, sdate.Second, DateTimeKind.Utc)).TotalSeconds;
}
}

```

```
// DELETE: api/ApiWithActions
[HttpDelete("{id:int}")]
[Route("Orders/{id:int}")]
public object Delete(int id)
{
    //handle delete action
    tree.Remove(tree.Where(or => or.taskId == id).FirstOrDefault());
    return Json(id);
}

public class CRUDModel<T> where T : class
{
    public TreeData Value;
    public int Key { get; set; }
    public int RelationalKey { get; set; }
    public List<TreeData> added { get; set; }
    public List<TreeData> changed { get; set; }
    public List<TreeData> deleted { get; set; }
}

public class TreeData
{
    public static List<TreeData> tree = new List<TreeData>();
    [System.ComponentModel.DataAnnotations.Key]
    public int taskId { get; set; }
    public string taskName { get; set; }
    public DateTime startDate { get; set; }
    public DateTime endDate { get; set; }
    public string duration { get; set; }
    public int progress { get; set; }
    public int? parentID { get; set; }
    public string predecessor { get; set; }
    public bool? isParent { get; set; }
    public bool? IsExpanded { get; set; }
    public static List<TreeData> GetTree()
```

```
{
if (tree.Count == 0)
{
Random rand = new Random();
var x = 0;
int duration = 0;
DateTime startDate = new DateTime(2000, 1, 3, 08, 00, 00);
for (var i = 1; i <= 50; i++)
{
startDate = startDate.AddDays(i == 1 ? 0 : 7);
DateTime childStartDate = startDate;
TreeData Parent = new TreeData()
{
taskId = ++x,
taskName = "Task " + x,
startDate = startDate,
endDate = childStartDate.AddDays(26),
duration = "20",
progress = rand.Next(100),
predecessor = null,
isParent = true,
parentID = null,
IsExpanded = false
};
tree.Add(Parent);
for (var j = 1; j <= 4; j++)
{
childStartDate = childStartDate.AddDays(j == 1 ? 0 : duration + 2);
duration = 5;
tree.Add(new TreeData()
{
taskId = ++x,
taskName = "Task " + x,
```

```

startDate = childStartDate,
endDate = childStartDate.AddDays(5),
duration = duration.ToString(),
progress = rand.Next(100),
parentID = Parent.taskId,
predecessor = (j > 1 ? (x - 1) + "FS" : ""),
isParent = false,
IsExpanded = false
});
}
}
}
return tree;
}
}
,

```

Limitations

- Filtering, sorting and searching are not supported in load on demand.
- Only Self-Referential type data is supported with remote data binding in Gantt Chart.
- Load-on-demand supports only the validated data source

Sending additional parameters to the server

We can pass additional parameters using [addParams](#) method of [Query](#) class.

In server side we have inherited and shown the additional parameter value in Syncfusion DataManager class itself. We pass an additional parameter in load time using [load](#) event. We can also pass additional parameter to the CRUD model. Please Check the below code snippet to send additional parameter to Gantt.

```

`ts
import { Gantt, Edit, Toolbar } from '@syncfusion/ej2-gantt';
import { DataManager, UrlAdaptor, Query } from '@syncfusion/ej2-data';
Gantt.Inject(Edit, Toolbar);
let dataSource: DataManager = new DataManager({
url: 'http://localhost:50039/Home/UrlDatasource',
adaptor: new UrlAdaptor,
batchUrl: 'http://localhost:50039/Home/BatchSave',

```



```

});
let gantt: Gantt = new Gantt({
  dataSource: dataSource,
  height: '450px',
  treeColumnIndex: 1,
  taskFields: {
    id: 'taskID',
    name: 'taskName',
    startDate: 'startDate',
    endDate: 'endDate',
    duration: 'duration',
    progress: 'progress',
    parentID: 'parentID',
  },
  editSettings: {
    allowAdding: true,
    allowEditing: true,
    allowDeleting: true,
  },
  toolbar: ['Add', 'Edit', 'Update', 'Delete', 'Cancel', 'ExpandAll', 'CollapseAll'],
  load: function (args) {
    this.query = new Query().addParams('ej2Gantt', "test");
  }
});
gantt.appendTo('#Gantt');
`ts
namespace URLAdaptor.Controllers
{
  public class HomeController : Controller
  {
    ...///
    //inherit the class to show age as property of DataManager

```

```

public class Test : DataManagerRequest
{
    public string ej2Gantt { get; set; }
}

public ActionResult UrlDatasource([FromBody]Test dm)
{
    if (DataList == null)
    {
        ProjectData datasource = new ProjectData();
        DataList = datasource.GetUrlDataSource();
    }
    var count = DataList.Count();
    return Json(new { result = DataList, count = count }, JsonRequestBehavior.AllowGet);
}

...///

public class ICRUDModel<T> where T : class
{
    public object key { get; set; }
    public T value { get; set; }
    public List<T> added { get; set; }
    public List<T> changed { get; set; }
    public List<T> deleted { get; set; }
    public IDictionary<string, object> @params { get; set; }
}

...///
}
}
,

```

You can find the full sample from [here](#).

Handling HTTP error

During server interaction from the Gantt, some server-side exceptions may occur, and you can acquire those error messages or exception details in client-side using the [actionFailure](#) event.

The argument passed to the `actionFailure` event contains the error details returned from the server.

INDEX.JS

```

var dataSource = new ej.data.DataManager({
    url: 'http://some.com/invalidUrl',
});
var gantt = new ej.gantt.Gantt({
    dataSource: dataSource,
    height: '450px',
    treeColumnIndex: 1,
    taskFields: {
        id: 'taskID',
        name: 'taskName',
        startDate: 'startDate',
        endDate: 'endDate',
        duration: 'duration',
        progress: 'progress',
        parentID: 'parentID',
    },
    actionFailure: (e) => {
        let span = document.createElement('span');
        gantt.element.parentNode.insertBefore(span, gantt.element);
        span.style.color = '#FF0000';
        span.innerHTML = 'Server exception: 404 Not found';
    },
});
gantt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>

```

```
</body></html>
```

Binding with Ajax

You can use Gantt [dataSource](#) property to bind the data source to Gantt from external Ajax request. In the below code we have fetched the data source from the server with the help of Ajax request and provided that to `dataSource` property by using [onSuccess](#) event of the Ajax.

INDEX.JS

```
var gantt = new ej.gantt.Gantt({
  height: '450px',
  treeColumnIndex: 1,
  taskFields: {
    id: 'TaskId',
    name: 'TaskName',
    startDate: 'StartDate',
    progress: 'Progress',
    duration: 'Duration',
    dependency: 'Predecessor',
    child: 'SubTasks'
  },
  projectStartDate: new Date('02/24/2019'),
  projectEndDate: new Date('07/20/2019')
});

gantt.appendTo('#Gantt');
let button = document.createElement('button');
button.textContent = 'Bind Data';
gantt.element.parentNode.insertBefore(button, gantt.element);
button.addEventListener("click", function(e) {
  let ajax = new
ej.base.Ajax("https://ej2services.syncfusion.com/production/web-
services/api/GanttData", "GET");
  gantt.showSpinner();
  ajax.send();
  ajax.onSuccess = function (data) {
    gantt.hideSpinner();
    gantt.dataSource = (JSON.parse(data)).Items;
  };
});
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Note: If you bind the dataSource from this way, then it acts like a local dataSource. So you cannot perform any server side crud actions.

Split task

The **Split-task** feature allows you to split a task or interrupt the work during planned or unforeseen circumstances. We can split the task either in load time or dynamically, by defining the segments either in hierarchical or self-referential way.

Hierarchical

To split a task at load time in hierarchical way, we need to define the segment details in datasource and this field should be mapped by using the [taskFields.segments](#) property.

```
`ts
```

```
[
{
```

```
TaskID: 1, TaskName: 'Identify Site location', StartDate: new Date('04/02/2019'), Duration: 4, Progress:
50,
```

```
Segments: [
```

```
{ StartDate: new Date("04/02/2019"), Duration: 2 },
```

```
{ StartDate: new Date("04/04/2019"), Duration: 2 }
```

```
]
```

```
}
```

```
]
```

```
,
```

INDEX.JS

```
var ganttChart = new ej.gantt.Gantt({
```

```

        dataSource: GanttData,
        height: "450px",
        taskFields: {
            id: "TaskID",
            name: "TaskName",
            startDate: "StartDate",
            endDate: "EndDate",
            duration: "Duration",
            progress: "Progress",
            child: "subtasks",
            segments: "Segments"
        },
    });
    ganttChart.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Self-referential

We can also define segment details as a flat data and this collection can be mapped by using [segmentData](#) property. The segment id field of this collection is mapped by using the [taskFields.segmentId](#) property.

`ts

```
taskFields: {
segmentId: "segmentId"
},
segmentData: [
{ segmentId: 1, StartDate: new Date("02/04/2019"), Duration: 2 },
{ segmentId: 1, StartDate: new Date("02/05/2019"), Duration: 5 },
{ segmentId: 4, StartDate: new Date("04/02/2019"), Duration: 2 },
{ segmentId: 4, StartDate: new Date("04/04/2019"), Duration: 2 }
],
`
```

INDEX.JS

```
var ganttChart = new ej.gantt.Gantt({
  dataSource: SplitTaskData,
  height: "450px",
  taskFields: {
    id: "TaskID",
    name: "TaskName",
    startDate: "StartDate",
    endDate: "EndDate",
    duration: "Duration",
    progress: "Progress",
    child: "subtasks",
    segmentId: "segmentId"
  },
  segmentData: [
    { segmentId: 2, StartDate: new Date("04/02/2019"), Duration: 2 },
    { segmentId: 2, StartDate: new Date("04/04/2019"), Duration: 2 },
    { segmentId: 4, StartDate: new Date("04/02/2019"), Duration: 2 },
    { segmentId: 4, StartDate: new Date("04/04/2019"), Duration: 2 }
  ],
});
ganttChart.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
```

```

<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Note: Segment id field contains id of a task which should be splitted at load time.

Limitations

Gantt has the support for both Hierarchical and Self-Referential data binding. When rendering the Gantt control with SQL database, we suggest you to use the Self-Referential data binding to maintain the parent-child relation. Because the complex json structure is very difficult to manage it in SQL tables, we need to write a complex queries and we have to write a complex algorithm to find out the proper record details while updating/deleting the inner level task in Gantt data source. We cannot implement both data binding for Gantt control and this is not a recommended way. If both child and parentID are mapped, the records will not render properly because, when task id of a record defined in the hierarchy structure is assigned to parent id of another record, in such case the records will not properly render. As the self-referential will search the record with particular id in flat data only, not in the inner level of records. If we map the parentID field, it will be prioritized and Gantt will be rendered based on the parentID values.

Immutable in ##Platform_Name## Gantt control

To enable this feature, you have to set the [enableImmutableMode](#) property as **true**.

This feature uses the primary key value for data comparison. So, you need to provide the [isPrimaryKey](#) column.

INDEX.JS

```

ej.gantt.Gantt.Inject(ej.gantt.DayMarkers);
var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    enableImmutableMode: true,
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },

```



```

        toolbar: ['Add', 'Edit', 'Delete', 'Update',
        'Cancel', 'Indent', 'Outdent'],
        editSettings: {
            allowAdding: true,
            allowEditing: true,
            allowDeleting: true,
            allowTaskbarEditing: true,
            showDeleteConfirmDialog: true
        }
    });
    ganttChart.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
    rel="stylesheet" type="text/css">

    <style>
        .e-gantt .e-gantt-chart .e-custom-holiday {
            background-color: #e82869;
        }
    </style>
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

Limitations

The following features are not supported in the immutable mode:

- Column reorder
- Virtualization

Virtual scroll in ##Platform_Name## Gantt control

Virtual Scroll support in Gantt allows you to load large amount of data without performance degradation. To enable Virtual Scrolling, you need to inject `VirtualScroll` module in Gantt.

Row virtualization

Row virtualization allows you to load and render a large number of tasks in Gantt with effective performance. In this mode, all tasks are fetched initially from the datasource and rendered in the DOM within a compact viewport area.

The number of records displayed in the Gantt is determined by the height.

This mode can be enable by setting the `enableVirtualization` property to `true`.

INDEX.JS

```
ej.gantt.Gantt.Inject(ej.gantt.Selection);
var ganttChart = new ej.gantt.Gantt({
  dataSource: virtualData,
  treeColumnIndex: 1,
  height: '450px',
  allowSelection: true,
  highlightWeekends: true,
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    endDate: 'EndDate',
    duration: 'Duration',
    progress: 'Progress',
    parentID: 'parentID'
  },
  enableVirtualization: true,
  columns: [
    { field: 'TaskID' },
    { field: 'TaskName' },
    { field: 'StartDate' },
    { field: 'Duration' },
    { field: 'Progress' }
  ],
  labelSettings: {
    taskLabel: 'Progress'
  },
  gridLines: 'Both',
  splitterSettings: {
    columnIndex: 2
  },
});
ganttChart.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
```

```

<title>EJ2 Gantt</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Gantt Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>

<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Timeline virtualization

Timeline virtualization allows you to load a data source having large timespan with high performance. Initially, it renders the timeline with thrice the width of the gantt element, while other timeline cells render on-demand during horizontal scrolling.

This mode can be enable by setting the [enableTimelineVirtualization](#) property to `true`.

INDEX.JS

```

ej.gantt.Gantt.Inject(ej.gantt.Selection);
var ganttChart = new ej.gantt.Gantt({
    dataSource: virtualData,
    treeColumnIndex: 1,
    height: '450px',
    allowSelection: true,
    highlightWeekends: true,
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        endDate: 'EndDate',
        duration: 'Duration',
        progress: 'Progress',
        parentID: 'parentID'
    },
},

```

```

enableVirtualization: true,
enableTimelineVirtualization: true,
columns: [
    { field: 'TaskID' },
    { field: 'TaskName' },
    { field: 'StartDate' },
    { field: 'Duration' },
    { field: 'Progress' }
],
labelSettings: {
    taskLabel: 'Progress'
},
gridLines: 'Both',
splitterSettings: {
    columnIndex: 2
},
projectStartDate: new Date('04/01/2019'),
projectEndDate: new Date('12/31/2025')
});
gantChart.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>

    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

Get filtered data when virtual scrolling is enabled

While enabling virtual scroll you can get the filtered or sorted record count using `filteredResult` from the `filterModule` of the treegrid inside the `actionComplete` event.

INDEX.JS

```
ej.gantt.Gantt.Inject(ej.gantt.Selection,ej.gantt.Sort,ej.gantt.Filter);
var filteredCount = 0;
var datasetCount;
var ganttChart = new ej.gantt.Gantt({
    dataSource: virtualData,
    treeColumnIndex: 1,
    height: '450px',
    allowSelection: true,
    highlightWeekends: true,
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        endDate: 'EndDate',
        duration: 'Duration',
        progress: 'Progress',
        parentID: 'parentID'
    },
    enableVirtualization: true,
    allowSorting: true,
    allowFiltering: true,
    actionComplete: function (args) {
        if (args.requestType == 'filtering') {
            if (args.rows != null) {
                filteredCount =
                    ganttChart.treeGrid.filterModule.filteredResult.length;
                var combinedMessage = `Dataset Count: ${datasetCount}
Filtered Count: ${filteredCount}`;
                var countElement = document.getElementById('count-
element');
                if (countElement) {
                    countElement.textContent = combinedMessage;
                }
            }
        }
    },
    created: function() {
        datasetCount = ganttChart.flatData.length;
        var combinedMessage = `Dataset Count: ${datasetCount} Filtered
Count: ${filteredCount}`;
        var countElement = document.getElementById('count-element');
        if (countElement) {
            countElement.textContent = combinedMessage;
        }
    },
    columns: [
        { field: 'TaskID' },
        { field: 'TaskName' },
        { field: 'StartDate' },
    ],
});
```

```

        { field: 'Duration' },
        { field: 'Progress' }
    ],
    labelSettings: {
        taskLabel: 'Progress'
    },
    gridLines: 'Both',
    splitterSettings: {
        columnIndex: 2
    },
    });
ganttChart.appendTo('#Gantt');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
    <h1>Counts Display</h1>
    <p id="count-element"></p>
  </div>

  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Limitations for virtual vcroll

- Due to the element height limitation in browsers, the maximum number of records loaded is limited by the browser capacity.
- Cell selection will not be persisted.

- The number of records rendered will be determined by the `height` property.
- It is necessary to mention the height of the Gantt in pixels when enabling Virtual Scrolling.
- Virtual Scroll does not support Multi Taskbar support in Resource View.

Selection

Selection in `##Platform_Name##` Gantt control

Selection provides an option to highlight a row or a cell. It can be done using arrow keys or by scrolling down the mouse. To disable selection in the Gantt control, set the [allowSelection](#) to false.

To select data, inject the [Selection](#) module into the Gantt control.

The Gantt control supports two types of selection that can be set by using the [selectionSettings.type](#) property. They are:

- **Single:** Sets a single value by default and allows only selection of a single row or a cell.
- **Multiple:** Allows you to select multiple rows or cells. To perform the multi-selection, press and hold the CTRL key and click the desired rows or cells.

Selection mode

The Gantt control supports three types of selection modes that can be set by using the [selectionSettings.mode](#). They are:

- **Row:** Allows you to select only rows, and the row value is set by default.
- **Cell:** Allows you to select only cells.
- **Both:** Allows you to select rows and cells at the same time.

INDEX.JS

```
ej.gantt.Gantt.Inject(ej.gantt.Selection);
var ganttChart = new ej.gantt.Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  selectionSettings: {
    mode: 'Both'
  }
});
ganttChart.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Gantt Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Toggle selection

The toggle selection allows you to select and deselect a specific row or cell. To enable toggle selection, set the `enableToggle` property of the `selectionSettings` to `true`. If you click the selected row or cell, then it will be deselected and vice versa. By default, the `enableToggle` property is set to `false`.

INDEX.JS

```

ej.gantt.Gantt.Inject(ej.gantt.Selection);
var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    selectionSettings: {
        mode: 'Row',
        type: 'Multiple',
        enableToggle: true
    }
});
ganttChart.appendTo('#Gantt');
var toggleBtn = new ej.buttons.Button();
toggleBtn.appendTo('#toggle');

```



```
document.getElementById('toggle').addEventListener('click', () => {
    gantt.selectionSettings.enableToggle = false;
});
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <button id="toggle">Disable toggle</button>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Clear selection

You can clear the selected cells and selected rows by using a method called [clearSelection](#). The following code example demonstrates how to clear the selected rows in Gantt Chart.

INDEX.JS

```
ej.gantt.Gantt.Inject(ej.gantt.Selection);
var ganttChart = new ej.gantt.Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  }
});
```

```

    },
    selectionSettings: {
        mode: 'Row',
        type: 'Multiple',
    }
});
gantChart.appendTo('#Gantt');
var selBtn= new ej.buttons.Button();
selBtn.appendTo('#selectRows');
let clrBtn = new ej.buttons.Button();
clrBtn.appendTo('#clearSelection');
document.getElementById('selectRows').addEventListener('click', () => {
    gantChart.selectionModule.selectRows([1,2,3]); // passing the
    record index as array collection
});
document.getElementById('clearSelection').addEventListener('click', () => {
    gantChart.clearSelection(); // Clear the selected rows
});

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <button id="selectRows">Select Rows</button>
    <button id="clearSelection">Clear Selection</button>

    <div id="container">
        <div id="Gantt"></div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

Get selected row indexes and records

You can get the selected row indexes by using the [getSelectedRowIndexes](#) method. And by using [getSelectedRecords](#) method, you can get the selected record details.

INDEX.JS

```
ej.gantt.Gantt.Inject(ej.gantt.Selection);
var ganttChart = new ej.gantt.Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  selectionSettings: {
    mode: 'Row',
    type: 'Multiple'
  },
  rowSelected: rowSelected
});
ganttChart.appendTo('#Gantt');
function rowSelected(args) {
  var selectedrowindex =
ganttChart.selectionModule.getSelectedRowIndexes(); // get the selected row
indexes.
  alert(selectedrowindex); // to alert the selected row indexes.
  var selectedrecords = ganttChart.selectionModule.getSelectedRecords();
// get the selected records.
  console.log(selectedrecords); // to print the selected records in
console window.
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
```

```

    <div id="container">
        <div id="Gantt"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Multiple Selection based on condition

You can select multiple rows based on condition by using the [selectRows](#) method.

In the following code, the rows which contains **TaskId** value as 3 and 4 are selected at initial rendering.

INDEX.JS

```

ej.gantt.Gantt.Inject(ej.gantt.Selection);
var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    selectionSettings: {
        mode: 'Row',
        type: 'Multiple'
    },
    dataBound: function(args) {
        var rowIndexes = [];
        ganttChart.treeGrid.grid.dataSource.forEach((data, index) => {
            if (data.TaskID === 3 || data.TaskID === 4) {
                rowIndexes.push(index);
            }
        });
        ganttChart.selectRows(rowIndexes);
    }
});
ganttChart.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">

```

```

<link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Touch interaction](#)

Row selection in ##Platform_Name## Gantt control

The row selection in the Gantt control can be enabled or disabled using the [allowSelection](#) property. You can get the selected row object using the [getSelectedRecords](#) method. The following code example shows how to disable the row selection in Gantt.

INDEX.JS

```

ej.gantt.Gantt.Inject(ej.gantt.Selection);
var ganttChart = new ej.gantt.Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  allowSelection: false
});
ganttChart.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Row selection is the default type of Gantt selection mode.

Selecting a row on initial load

You can select a row at the time of loading by setting the index of the row to the [selectedRowIndex](#) property. Find the following code example for details.

INDEX.JS

```

ej.gantt.Gantt.Inject(ej.gantt.Selection);
var ganttChart = new ej.gantt.Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  selectedRowIndex: 3,
});
ganttChart.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Selecting a row dynamically

You can also select a row dynamically using the [selectRow](#) method. The following code demonstrates how to select a row dynamically by clicking the custom button.

INDEX.JS

```

ej.gantt.Gantt.Inject(ej.gantt.Selection);
var ganttChart = new ej.gantt.Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  }
});
ganttChart.appendTo('#Gantt');
var selectBtn= new ej.buttons.Button();
selectBtn.appendTo('#selectRow');
document.getElementById('selectRow').addEventListener('click', () => {

```

```
ganttChart.selectionModule.selectRow(2); // passing the record index to
select the row
});
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <button id="selectRow">Select Row</button>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Multiple row selection

You can select multiple rows by setting the [selectionSettings.type](#) property to `multiple`. You can select more than one row by holding down the CTRL key while selecting multiple rows. The following code example explains how to enable multiple selection in Gantt.

INDEX.JS

```
ej.gantt.Gantt.Inject(ej.gantt.Selection);
var ganttChart = new ej.gantt.Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
```



```

        child: 'subtasks'
    },
    selectionSettings: {
        mode: 'Row',
        type: 'Multiple'
    }
});
gantChart.appendTo('#Gantt');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Selecting multiple rows dynamically

You can also select rows dynamically using the [selectRows](#) method. The following code demonstrates how to select rows dynamically by clicking the custom button.

INDEX.JS

```

ej.gantt.Gantt.Inject(ej.gantt.Selection);
var ganttChart = new ej.gantt.Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
```

```

        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    selectionSettings: {
        mode: 'Row',
        type: 'Multiple',
    }
});
gantChart.appendTo('#Gantt');
var selBtn= new ej.buttons.Button();
selBtn.appendTo('#selectRows');
document.getElementById('selectRows').addEventListener('click', () => {
    gantChart.selectionModule.selectRows([1,2,3]); // passing the
    record index as array collection
});

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <button id="selectRows">Select Rows</button>

    <div id="container">
        <div id="Gantt"></div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

Customize row selection action

While selecting a row in Gantt, the [rowSelecting](#) and [rowSelected](#) events will be triggered. The [rowSelecting](#) event will be triggered on initialization of row selection, and you can get the previously

selected row and current selecting row's information, which is used to prevent selection of a particular row. The [rowSelected](#) event will be triggered on completion of row selection action, and you can get the current selected row's information through this event. The following code example demonstrates how to prevent the selection of a row using the [rowSelecting](#) event.

INDEX.JS

```
ej.gantt.Gantt.Inject(ej.gantt.Selection);
var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    rowSelecting: function(args) {
        if (args.data.TaskID==4) {
            args.cancel=true;
        }
    },
    selectionSettings: {
        mode: 'Row'
    }
});
ganttChart.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
```

```

if(ele) {
    ele.style.visibility = "visible";
}

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

In the Gantt control, when you click an already selected row, selection will be cleared. While deselecting a row in Gantt, the [rowDeselecting](#) and [rowDeselected](#) events will occur. The [rowDeselecting](#) event will occur on initialization of deselecting row, and you can get the current deselecting row's information to prevent the deselection of particular row. The [rowDeselected](#) event will occur on completion of row deselection action, and you can get the current deselected row's information.

Cell selection in ##Platform_Name## Gantt control

You can select a cell in the Gantt control by setting the [selectionSettings.mode](#) property to cell. You can get the selected cell information using the [getSelectedRowCellIndexes](#) method. This method returns the result as an object collection, which has [cellIndexes](#) and [rowIndex](#) information of the selected cells.

Find the code example below to enable the cell selection in Gantt.

INDEX.JS

```

ej.gantt.Gantt.Inject(ej.gantt.Selection);
var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    selectionSettings: {
        mode: 'Cell'
    }
});
ganttChart.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>

```

```

<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Selecting multiple cells

You can select multiple cells by setting the [selectionSettings.type](#) property to multiple and the [selectionSettings.mode](#) property to cell. Multiple cells can be selected by holding the CTRL key and selecting the cells. The following code example demonstrates how to select multiple cells.

INDEX.JS

```

ej.gantt.Gantt.Inject(ej.gantt.Selection);
var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    selectionSettings: {
        mode: 'Cell',
        type: 'Multiple'
    }
});
ganttChart.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Selecting a cell dynamically

You can select a cell dynamically using the [selectCell](#) method. Refer to the following code example for details.

INDEX.JS

```

ej.gantt.Gantt.Inject(ej.gantt.Selection);
var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    selectionSettings: {
        mode: 'Cell'
    }
});
ganttChart.appendTo('#Gantt');
var cellBtn= new ej.buttons.Button();
cellBtn.appendTo('#selectCell');
document.getElementById('selectCell').addEventListener('click', () => {
    ganttChart.selectionModule.selectCell({ cellIndex: 1, rowIndex: 1
});
});

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>

```

```

<title>EJ2 Gantt</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Gantt Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <button id="selectCell">Select Cell</button>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customize cell selection action

While selecting a cell in Gantt, the [cellSelecting](#) and [cellSelected](#) event will be triggered. The [cellSelecting](#) event will be triggered on initialization of cell selection action, and you can get the current selecting cell information to prevent the selection of a particular cell in a particular row. The [cellSelected](#) event will be triggered on completion of cell selection action, and you can get the current selected cell's information. The following code example demonstrates how to prevent the selection of the cell using the [cellSelecting](#) event.

INDEX.JS

```

ej.gantt.Gantt.Inject(ej.gantt.Selection);
var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    cellSelecting: function(args) {
        if (args.data.TaskID==4 && args.cellIndex.cellIndex==1) {

```

```

        args.cancel=true;
    }
    },
    selectionSettings: {
        mode: 'Cell'
    }
});
gantChart.appendTo('#Gantt');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Filtering

Filtering in ##Platform_Name## Gantt control

Filtering allows you to view specific or related records based on filter criteria. This can be done in the Gantt control by using the filter menu support and toolbar search support. To enable filtering in the Gantt control, set the [allowFiltering](#) to true. Menu filtering support can be configured using the [filterSettings](#) property and toolbar searching can be configured using the [searchSettings](#) property.

To use filter, inject the [Filter](#) module into the Gantt control.

Filter hierarchy modes

The Gantt supports a set of filtering modes with the [filterSettings.hierarchyMode](#) property. The following are the types of filter hierarchy modes available in the Gantt control:

- **Parent:** This is the default filter hierarchy mode in Gantt. The filtered records are displayed with its parent records. If the filtered records do not have any parent record, then only the filtered records will be displayed.
- **Child:** Displays the filtered records with its child record. If the filtered records do not have any child record, then only the filtered records will be displayed.
- **Both:** Displays the filtered records with its both parent and child records. If the filtered records do not have any parent and child records, then only the filtered records will be displayed.
- **None:** Displays only the filtered records.

INDEX.JS

```
ej.gantt.Gantt.Inject(ej.gantt.Filter);
var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    allowFiltering: true
});
ganttChart.appendTo('#Gantt');
var dropDownMode = new ej.dropdowns.DropDownList({
    dataSource: [
        { id: 'Parent', mode: 'Parent' },
        { id: 'Child', mode: 'Child' },
        { id: 'Both', mode: 'Both' },
        { id: 'None', mode: 'None' },
    ],
    fields: { text: 'mode', value: 'id' },
    value: 'Parent',
    change: function (e) {
        var mode = e.value;
        ganttChart.filterSettings.hierarchyMode = mode;
        ganttChart.clearFiltering();
    }
});
dropDownMode.appendTo('#mode');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div style="padding-top: 7px; display: inline-block">Hierarchy
Mode</div>
            <div style="display: inline-block">
                <input type="text" id="mode">
            </div>
        </div>
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Initial filter

To apply the filter at initial rendering, set the filter to `predicate` object in the `filterSettings.columns` property.

INDEX.JS

```

ej.gantt.Gantt.Inject(ej.gantt.Filter);
var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    filterSettings: {
        columns: [{ field: 'TaskName', matchCase: false, operator:
'startswith', predicate: 'and', value: 'Identify' },
        { field: 'TaskID', matchCase: false, operator: 'equal',
predicate: 'and', value: 2 } ]
    },
    allowFiltering: true
});

```

```
ganttChart.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Filter operators

The filter operator for a column can be defined in the `filterSettings.columns.operator` property.

The available operators and its supported data types are:

Operator	Description	Supported Types
startswith	Checks whether the value begins with the specified value.	String
endswith	Checks whether the value ends with the specified value.	String
contains	Checks whether the value contains the specified value.	String
equal	Checks whether the value is equal to the specified value.	String | Number | Boolean | Date
notequal	Checks for the values that are not equal to the specified value.	String | Number | Boolean | Date
greaterthan	Checks whether the value is greater than the specified value.	Number | Date

greaterthanorequal | Checks whether the value is greater than or equal to the specified value. | Number | Date

lessthan | Checks whether the value is less than the specified value. | Number | Date

lessthanorequal | Checks whether the value is less than or equal to the specified value. | Number | Date

By default, the `filterSettings.columns.operator` value is **equal**

Diacritics

By default, the Gantt control ignores the diacritic characters while filtering. To include diacritic characters, set the `filterSettings.ignoreAccent` to true.

In the following sample, type **Perform** in the **TaskName** column to filter diacritic characters.

INDEX.JS

```
ej.gantt.Gantt.Inject(ej.gantt.Filter);
var GanttData = [
    {
        TaskID: 1,
        TaskName: 'Project initiation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        subtasks: [
            {TaskID: 2, TaskName: 'Identify site location', StartDate: new
Date('04/02/2019'), Duration: 0, Progress: 50 },
            {TaskID: 3, TaskName: 'Perform soil test', StartDate: new
Date('04/02/2019'), Duration: 4, Progress: 50 },
            {TaskID: 4, TaskName: 'Soil test approval', StartDate: new
Date('04/02/2019'), Duration: 0, Progress: 50 },
        ]
    },
    {
        TaskID: 5,
        TaskName: 'Project estimation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        subtasks: [
            {TaskID: 6, TaskName: 'Develop floor plan for estimation',
StartDate: new Date('04/04/2019'), Duration: 3, Progress: 50, resources:
[4]},
            {TaskID: 7, TaskName: 'List materials', StartDate: new
Date('04/04/2019'), Duration: 3, Progress: 50},
            {TaskID: 8, TaskName: 'Estimation approval', StartDate: new
Date('04/04/2019'), Duration: 0, Progress: 50 }
        ]
    }
];

var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
```

```

        progress: 'Progress',
        child: 'subtasks'
    },
    filterSettings: {
        ignoreAccent: true
    },
    allowFiltering: true
});
gantChart.appendTo('#Gantt');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Filtering a specific column by method

You can filter the columns dynamically by using the [filterByColumn](#) method.

INDEX.JS

```

ej.gantt.Gantt.Inject(ej.gantt.Filter);
var ganttChart = new ej.gantt.Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
```

```

        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    allowFiltering: true
});
gantChart.appendTo('#Gantt');
var filterBtn= new ej.buttons.Button();
filterBtn.appendTo('#filter');
document.getElementById('filter').addEventListener('click', function() {
    gantChart.filterByColumn('TaskName','startswith','Iden');
});

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">

        <button id="filter">Filter TaskName column</button>
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Clear filtered columns

You can clear all the filtering condition done in the Gantt control by using the [clearFiltering](#) method.

The following code snippet explains the above behavior.

INDEX.JS

```

ej.gantt.Gantt.Inject(ej.gantt.Filter);
var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    allowFiltering: true,
    filterSettings: {
        columns: [{ field: 'TaskName', matchCase: false, operator:
'startswith', predicate: 'and', value: 'Identify' },
        { field: 'Progress', matchCase: false, operator: 'equal',
predicate: 'and', value: 50 }]}
    });
ganttChart.appendTo('#Gantt');
var filterBtn = new ej.buttons.Button();
filterBtn.appendTo('#clearFilter');
document.getElementById('clearFilter').addEventListener('click', function()
{
    ganttChart.clearFiltering();
});

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="clearFilter">Clear Filter</button>
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');

```

```

if(ele) {
    ele.style.visibility = "visible";
}

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Filter menu in ##Platform_Name## Gantt control

The Gantt control provides the menu filtering support for each column. You can enable the filter menu by setting the [allowFiltering](#) to `true`. The filter menu UI will be rendered based on its column type, which allows you to filter data. You can filter the records with different operators.

INDEX.JS

```

ej.gantt.Gantt.Inject(ej.gantt.Filter);
var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    filterSettings: {
        type: 'Menu'
    },
    allowFiltering: true
});
ganttChart.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```



```

    <div id="container">
        <div id="Gantt"></div>
    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The [allowFiltering](#) property should be set to **true** to enable the filter menu.

Setting the [columns.allowFiltering](#) property to **false** prevents rendering filter menu for a particular column.

Custom component in filter menu

The [column.filter.ui](#) is used to add custom filter components to a particular column.

To implement custom filter ui, define the following functions:

- **create**: Creates custom component.
- **write**: Wire events for custom component.
- **read**: Read the filter value from custom component.

In the following sample, dropdown is used as custom component in the TaskName column.

INDEX.JS

```

ej.gantt.Gantt.Inject(ej.gantt.Filter);
var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    columns: [
        { field: 'TaskID' },
        { field: 'TaskName', filter: {
            ui: {
                create: function(args) {
                    var db = new
ej.data.DataManager(ganttChart.treeGrid.grid.dataSource);
                    var flValInput = ej.base.createElement('input', {
className: 'flm-input' });
                    args.target.appendChild(flValInput);
                    this.dropInstance = new ej.dropdowns.DropDownList({
                        dataSource: new
ej.data.DataManager(ganttChart.treeGrid.grid.dataSource),

```

```

        fields: { text: 'TaskName', value: 'TaskName' },
        placeholder: 'Select a value',
        popupHeight: '200px'
    });
    this.dropInstance.appendTo(flValInput);
},
write: function(args){
    this.dropInstance.value = args.filteredValue;
},
read: function(args) {
    args.fltrObj.filterByColumn(args.column.field,
args.operator, this.dropInstance.value);
}
}
},
{ field: 'StartDate' },
{ field: 'Duration' }
],
allowFiltering: true
});
gantChart.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Excel like filter in ##Platform_Name## Gantt control

You can enable Excel like filter by defining [filterSettings.type](#) as **Excel**. The excel menu contains an option such as Sorting, Clear filter, Sub menu for advanced filtering.

INDEX.JS

```
ej.gantt.Gantt.Inject(ej.gantt.Filter);
var ganttChart = new ej.gantt.Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  filterSettings: {
    type: 'Menu'
  },
  allowFiltering: true
});
ganttChart.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
```

```

}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Searching in ##Platform_Name## Gantt control

You can search records in the Gantt control by using the [search](#) method with search key as a parameter. The Gantt control provides an option to integrate the search text box in the toolbar by adding the search item to the [toolbar](#) property.

To search records, inject the **Filter** module into the Gantt control.

INDEX.JS

```

ej.gantt.Gantt.Inject(ej.gantt.Filter, ej.gantt.Toolbar);
var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    toolbar: ['Search']
});
ganttChart.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>

```

```
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Initial search

In the Gantt control, you can load a task with some search criteria and this can be done by using the [searchSettings](#) property. To apply search at initial rendering, set the value for [fields](#), [operator](#), [key](#), and [ignoreCase](#) in the [searchSettings](#) property.

INDEX.JS

```
ej.gantt.Gantt.Inject(ej.gantt.Filter, ej.gantt.Toolbar);
var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    toolbar: ['Search'],
    searchSettings: { fields: ['TaskName'], operator:
'contains', key: 'List', ignoreCase: true }
});
ganttChart.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
```

```

    <div id="container">
      <div id="Gantt"></div>
    </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

By default, Gantt searches all the bound column values. To customize this behavior, define the [searchSettings.fields](#) property.

Search operators

The search operator can be defined in the [searchSettings.operator](#) property to configure specific searching.

The following operators are supported in searching:

Operator | Description

startsWith | Checks whether a value begins with the specified value.

endsWith | Checks whether a value ends with the specified value.

contains | Checks whether a value contains the specified value.

equal | Checks whether a value is equal to the specified value.

notEqual | Checks for the values that are not equal to the specified value.

By default, the [searchSettings.operator](#) value is `contains`.

Search by external button

To search the Gantt records from an external button, invoke the [search](#) method.

INDEX.JS

```

ej.gantt.Gantt.Inject(ej.gantt.Filter, ej.gantt.Toolbar);
var ganttChart = new ej.gantt.Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  allowFiltering: true
});
ganttChart.appendTo('#Gantt');
var searchBtn= new ej.buttons.Button();
searchBtn.appendTo('#search');

```

```
document.getElementById('search').addEventListener('click', function() {
    var searchText = document.getElementsByClassName('searchtext')[0].value;
    ganttChart.search(searchText);
});
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="e-float-input" style="width: 200px; display:
inline-block;">
            <input type="text" class="searchtext">
            <span class="e-float-line"></span>
            <label class="e-float-text">Search text</label>
        </div>
        <button id="search">Search</button>
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Note: You should set the [allowFiltering](#) property to `true` for searching the content externally.

Search specific columns

By default, the Gantt control searches all the columns. You can search specific columns by defining the specific column's field names in the [searchSettings.fields](#) property.

INDEX.JS

```
ej.gantt.Gantt.Inject(ej.gantt.Filter, ej.gantt.Toolbar);
var ganttChart = new ej.gantt.Gantt({
```

```

        dataSource: GanttData,
        height: '450px',
        columns: [
            { field: 'TaskID', headerText: 'Task ID', textAlign: 'Left',
width: '100' },
            { field: 'TaskName', headerText: 'Task Name', width: '200' },
            { field: 'StartDate', headerText: 'Start Date', width: '150' },
            { field: 'Duration', headerText: 'Duration', width: '150' },
            { field: 'Progress', headerText: 'Progress', width: '150' },
        ],
        taskFields: {
            id: 'TaskID',
            name: 'TaskName',
            startDate: 'StartDate',
            duration: 'Duration',
            progress: 'Progress',
            child: 'subtasks'
        },
        toolbar: ['Search'],
        searchSettings: { fields: ['TaskName', 'Duration']}
    });
    ganttChart.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
<script src="index.js" type="text/javascript"></script>

```



```
</body></html>
```

In above sample, you can search only **TaskName** and **Duration** column values.

Clear search by external button

You can set [searchSettings.key](#) property as **empty** string, to clear the searched Gantt records from external button.

INDEX.JS

```
ej.gantt.Gantt.Inject(ej.gantt.Filter, ej.gantt.Toolbar);
var ganttChart = new ej.gantt.Gantt({
  dataSource: GanttData,
  height: '450px',
  columns: [
    { field: 'TaskID', headerText: 'Task ID', textAlign: 'Left', width:
'100' },
    { field: 'TaskName', headerText: 'Task Name', width: '250' },
    { field: 'StartDate', headerText: 'Start Date', width: '150' },
    { field: 'Duration', headerText: 'Duration', width: '150' },
    { field: 'Progress', headerText: 'Progress', width: '150' },
  ],
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  toolbar: ['Search'],
  searchSettings: { fields: ['TaskName'], operator: 'contains', key:
'Perform', ignoreCase: true },
});
ganttChart.appendTo('#Gantt');
var clearBtn= new ej.buttons.Button();
clearBtn.appendTo('#changeByPosition');
document.getElementById('clear').addEventListener('click', function() {
  ganttChart.searchSettings.key='';
});
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
```

```

<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="clear">Clear Search</button>
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Task dependency in ##Platform_Name## Gantt control

Task dependency or task relationship can be established between two tasks in Gantt. This dependency affects the project schedule. If you change the predecessor of a task, it will affect the successor task, which will affect the next task, and so on. Relationship can be established between parent-parent tasks, child-child tasks, parent-child and child-parent task.

In Gantt, you can enable or disable the parent predecessor using [allowParentDependency](#) property.

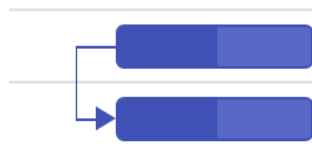
By default, the `allowParentDependency` property will be `true`.

Task relationship types

Task relationships are categorized into four types based on the start and finish dates of the task.

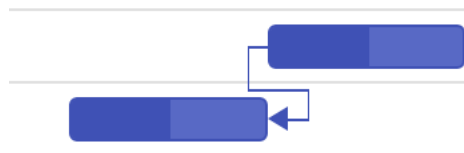
Start to Start (SS)

You cannot start a task until the dependent task also starts.



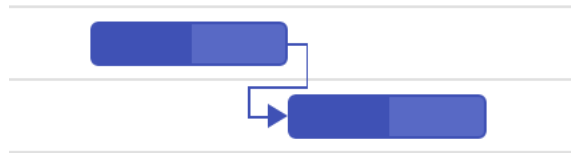
Start to Finish (SF)

You cannot finish a task until the dependent task is started.



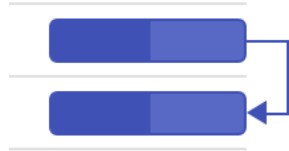
Finish to Start (FS)

You cannot start a task until the dependent task is completed.



Finish to Finish (FF)

You cannot finish a task until the dependent task is completed.



Define task relationship

Task relationship is defined in the data source as a string value, and this value is mapped to the Gantt control by using the [taskFields.dependency](#) property. The following code example demonstrates how to enable the predecessor in the Gantt control.

INDEX.JS

```
var GanttData = [
    {
        TaskID: 1,
        TaskName: 'Project Initiation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        subtasks: [
            { TaskID: 2, TaskName: 'Identify Site location', StartDate:
new Date('04/02/2019'), Duration: 0, Progress: 50 },
            { TaskID: 3, TaskName: 'Perform Soil test', StartDate: new
Date('04/02/2019'), Duration: 4, Progress: 50 },
            { TaskID: 4, TaskName: 'Soil test approval', StartDate: new
Date('04/02/2019'), Duration: 4,Predecessor:"2FS", Progress: 50 },
        ]
    },
    {
        TaskID: 5,
        TaskName: 'Project Estimation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        subtasks: [
            { TaskID: 6, TaskName: 'Develop floor plan for estimation',
StartDate: new Date('04/04/2019'), Duration: 3, Progress: 50 },
            { TaskID: 7, TaskName: 'List materials', StartDate: new
Date('04/04/2019'), Duration: 3, Progress: 50 },
            { TaskID: 8, TaskName: 'Estimation approval', StartDate: new
Date('04/04/2019'), Duration: 0,Predecessor:"6SS", Progress: 50 }
        ]
    },
];

var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
```

```

        name: 'TaskName',
        startDate: 'StartDate',
        dependency: 'Predecessor',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks',
    }
    });
    ganttChart.appendTo('#Gantt');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Predecessor offset with duration units

In the Gantt control, the predecessor offset can be defined with the following duration units:

- Day
- Hour
- Minute

You can define an offset with various offset duration units for predecessors by using the following code example.

INDEX.JS

```

var ganttData = [
    {
        TaskID: 1,
        TaskName: 'Project Initiation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        subtasks: [
            { TaskID: 2, TaskName: 'Identify Site location', StartDate:
new Date('04/02/2019'), Duration: 0, Progress: 50 },
            { TaskID: 3, TaskName: 'Perform Soil test', StartDate: new
Date('04/02/2019'), Duration: 4, Progress: 50 },
            { TaskID: 4, TaskName: 'Soil test approval', StartDate: new
Date('04/04/2019'), Duration: 4,Predecessor:"2FS+2days", Progress: 50 },
            { TaskID: 5, TaskName: 'Clear the building
site', StartDate: new Date('04/04/2019'), Duration: 2, Progress: 30,
Predecessor: '4FF+960m'}
        ]
    },
    {
        TaskID: 6,
        TaskName: 'Project Estimation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        subtasks: [
            { TaskID: 7, TaskName: 'Develop floor plan for estimation',
StartDate: new Date('04/04/2019'), Duration: 3, Progress: 50 },
            { TaskID: 8, TaskName: 'List materials', StartDate: new
Date('04/04/2019'), Duration: 3, Progress: 50 },
            { TaskID: 9, TaskName: 'Estimation approval', StartDate: new
Date('04/06/2019'), Duration: 0,Predecessor:"7SS+16h", Progress: 50 }
        ]
    },
];

var ganttChart = new ej.gantt.Gantt({
    dataSource: ganttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        dependency: 'Predecessor',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks',
    },
    columns: [
        { field: 'TaskID', headerText: 'Task ID', textAlign: 'Left',
width: '100' },
        { field: 'Predecessor', headerText: 'Dedependency', width:
'150'},
        { field: 'TaskName', headerText: 'Task Name', width: '150' },
        { field: 'StartDate', headerText: 'Start Date',
width: '150' },
        { field: 'Duration', headerText: 'Duration', width: '150' },
    ]
});

```

```

        { field: 'Progress', headerText: 'Progress', width: '150' }
    ];
});
gantChart.appendTo('#Gantt');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Disabling automatic dependency offset updates

By default, the dependency offsets are automatically updated in the Gantt chart whenever a task's start or end date is changed. However, if you want to disable this feature, you can do so by disabling the [updateOffsetOnTaskbarEdit](#) property. Once this property is disabled, you can only update the offset value by editing the predecessor column cell or the offset column in the dependency tab of the edit dialog.

INDEX.JS

```

var ganttData = [
  {
    TaskID: 1,
    TaskName: 'Project Initiation',
    StartDate: new Date('04/02/2019'),
    EndDate: new Date('04/21/2019'),
    subtasks: [
```

```

        { TaskID: 2, TaskName: 'Identify Site location', StartDate:
new Date('04/02/2019'), Duration: 0, Progress: 50 },
        { TaskID: 3, TaskName: 'Perform Soil test', StartDate: new
Date('04/02/2019'), Duration: 4, Progress: 50 },
        { TaskID: 4, TaskName: 'Soil test approval', StartDate: new
Date('04/04/2019'), Duration: 4,Predecessor:"2FS+2days", Progress: 50 },
        { TaskID: 5, TaskName: 'Clear the building
site', StartDate: new Date('04/04/2019'), Duration: 2, Progress: 30,
Predecessor: '4FF+960m'}
    ]
},
{
    TaskID: 6,
    TaskName: 'Project Estimation',
    StartDate: new Date('04/02/2019'),
    EndDate: new Date('04/21/2019'),
    subtasks: [
        { TaskID: 7, TaskName: 'Develop floor plan for estimation',
StartDate: new Date('04/04/2019'), Duration: 3, Progress: 50 },
        { TaskID: 8, TaskName: 'List materials', StartDate: new
Date('04/04/2019'), Duration: 3, Progress: 50 },
        { TaskID: 9, TaskName: 'Estimation approval', StartDate: new
Date('04/06/2019'), Duration: 0,Predecessor:"7SS+16h", Progress: 50 }
    ]
},
];

var ganttChart = new ej.gantt.Gantt({
    dataSource: ganttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        dependency: 'Predecessor',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks',
    },
    columns: [
        { field: 'TaskID', headerText: 'Task ID', textAlign: 'Left',
width: '100' },
        { field: 'Predecessor', headerText: 'Depedency', width:
'150' },
        { field: 'TaskName', headerText: 'Task Name', width: '150' },
        { field: 'StartDate', headerText: 'Start Date',
width: '150' },
        { field: 'Duration', headerText: 'Duration', width: '150' },
        { field: 'Progress', headerText: 'Progress', width: '150' }
    ]
});
ganttChart.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
```

```

<title>EJ2 Gantt</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Gantt Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Validate predecessor links on editing

In Gantt, task relationship link can be broken by editing the start date, end date and duration value of task. When the task relationship broken on any edit action. This can be handled in Gantt in two ways.

- Using actionBegin event
- Using predecessor validation dialog

Using actionBegin event

When the task relationship link is broken on any edit action, then the [actionBegin](#) event will be triggered with `requestType` argument as `validateLinkedTask`. You can validate the editing action within the `actionBegin` event using the `validateMode` event argument. The `validateMode` event argument has the following properties:

Argument	Default value	Description
<code>args.validateMode.respectLink</code>	<code>false</code>	In this validation mode, the predecessor links get high priority. With this mode enabled, when the successor task is moved before the predecessor task's end date, the editing will be reverted, and dates will be validated based on the dependency links.
<code>args.validateMode.removeLink</code>	<code>false</code>	In this validation mode, the taskbar editing gets high priority. For inappropriate task dates, the dependency links will be removed and tasks will be moved to the edited date.

`args.validateMode.preserveLinkWithEditing | true` | In this validation mode, the taskbar editing will be considered along with the dependency links. This relationship will be maintained by updating the offset value of predecessors.

By default, the `preserveLinkWithEditing` validation mode will be enabled, so the predecessors are updated with offset values.

The following sample explains enabling the `respectLink` validation mode while editing the linked tasks in the [actionBegin](#) event.

INDEX.JS

```
var GanttData = [
    {
        TaskID: 1,
        TaskName: 'Project Initiation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        subtasks: [
            { TaskID: 2, TaskName: 'Identify Site location', StartDate:
new Date('04/02/2019'), Duration: 0, Progress: 50 },
            { TaskID: 3, TaskName: 'Perform Soil test', StartDate: new
Date('04/02/2019'), Duration: 4, Progress: 50 },
            { TaskID: 4, TaskName: 'Soil test approval', StartDate: new
Date('04/02/2019'), Duration: 4, Predecessor: "2FS", Progress: 50 },
        ]
    },
    {
        TaskID: 5,
        TaskName: 'Project Estimation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        subtasks: [
            { TaskID: 6, TaskName: 'Develop floor plan for estimation',
StartDate: new Date('04/04/2019'), Duration: 3, Progress: 50 },
            { TaskID: 7, TaskName: 'List materials', StartDate: new
Date('04/04/2019'), Duration: 3, Progress: 50 },
            { TaskID: 8, TaskName: 'Estimation approval', StartDate: new
Date('04/04/2019'), Duration: 0, Predecessor: "6SS", Progress: 50 }
        ]
    }
];

var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        dependency: 'Predecessor',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks',
    },
    editSettings: {
        allowTaskbarEditing: true
    }
});
```

```

        },
        actionBegin: function (args) {
            if (args.requestType == "validateLinkedTask") {
                args.validateMode.respectLink = true;
            }
        }
    });
    ganttChart.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Using validation dialog

When disabling all the validation modes in the [actionBegin](#) event, a validation pop-up will be displayed prompting users to select the validation mode to validate taskbar editing.

This validation pop-up will display different options based on the successor task's start date after editing.

If you move the successor task that starts after the predecessor task's end date, then a dialog will be rendered with the following options:

- Cancel, Keep the existing link.

- Remove the link and move the task to start on edited date.
- Move the task to start on edited date and keep the link.

If you move the successor task that starts before the predecessor task's end date, then a dialog will be rendered with the following options:

- Cancel, Keep the existing link.
- Remove the link and move the task to start on edited date.

The following code example shows how to enable the predecessor validation dialog in Gantt.

INDEX.JS

```
var GanttData = [
    {
        TaskID: 1,
        TaskName: 'Project Initiation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        subtasks: [
            { TaskID: 2, TaskName: 'Identify Site location', StartDate:
new Date('04/02/2019'), Duration: 0, Progress: 50 },
            { TaskID: 3, TaskName: 'Perform Soil test', StartDate: new
Date('04/02/2019'), Duration: 4, Progress: 50 },
            { TaskID: 4, TaskName: 'Soil test approval', StartDate: new
Date('04/02/2019'), Duration: 4,Predecessor:"2FS", Progress: 50 },
        ]
    },
    {
        TaskID: 5,
        TaskName: 'Project Estimation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        subtasks: [
            { TaskID: 6, TaskName: 'Develop floor plan for estimation',
StartDate: new Date('04/04/2019'), Duration: 3, Progress: 50 },
            { TaskID: 7, TaskName: 'List materials', StartDate: new
Date('04/04/2019'), Duration: 3, Progress: 50 },
            { TaskID: 8, TaskName: 'Estimation approval', StartDate: new
Date('04/04/2019'), Duration: 0,Predecessor:"6SS", Progress: 50 }
        ]
    },
];

var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        dependency: 'Predecessor',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    }
});
```

```

    },
    editSettings:{
        allowTaskbarEditing:true
    },
    actionBegin:function(args){
        if (args.requestType == "validateLinkedTask") {
            args.validateMode.preserveLinkWithEditing = false;
        }
    }
});
gantChart.appendTo('#Gantt');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Dynamically show/hide the dependency line

By default, mapping the dependency field in taskFields displays dependency lines in the Gantt chart. To hide the dependency line upon button click, set `visibility` style to hidden for the CSS class name `.e-gantt-dependency-view-container`.

INDEX.JS

```

var ganttData = [
  {
```

```

        TaskID: 1,
        TaskName: 'Project Initiation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        subtasks: [
            { TaskID: 2, TaskName: 'Identify Site location', StartDate: new
Date('04/02/2019'), Duration: 0, Progress: 50, },
            { TaskID: 3, TaskName: 'Perform Soil test', StartDate: new
Date('04/02/2019'), Duration: 4, Progress: 50, },
            { TaskID: 4, TaskName: 'Soil test approval', StartDate: new
Date('04/04/2019'), Duration: 4, Predecessor: '2FS+2days', Progress: 50, },
            { TaskID: 5, TaskName: 'Clear the building site', StartDate: new
Date('04/04/2019'), Duration: 2, Progress: 30, Predecessor: '4FF+960m', },
        ],
    },
    {
        TaskID: 6,
        TaskName: 'Project Estimation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        subtasks: [
            { TaskID: 7, TaskName: 'Develop floor plan for estimation',
StartDate: new Date('04/04/2019'), Duration: 3, Progress: 50, },
            { TaskID: 8, TaskName: 'List materials', StartDate: new
Date('04/04/2019'), Duration: 3, Progress: 50, },
            { TaskID: 9, TaskName: 'Estimation approval', StartDate: new
Date('04/06/2019'), Duration: 0, Predecessor: '7SS+16h', Progress: 50, },
        ],
    },
];

// Switch toggle.
var switchObj = new ej.buttons.Switch({ checked: false, change: Onchange,
});
switchObj.appendTo('#switch');
function Onchange() {
    var ganttDependencyViewContainer = document.querySelector('.e-gantt-
dependency-view-container');
    if (switchObj.checked) {
        ganttDependencyViewContainer.style.visibility = 'hidden';
    }
    else {
        ganttDependencyViewContainer.style.visibility = 'visible';
    }
}
var ganttChart = new ej.gantt.Gantt({
    dataSource: ganttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        dependency: 'Predecessor',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks',
    },
},

```

```

columns: [
  { field: 'TaskID', headerText: 'Task ID', textAlign: 'Left', width:
'100' },
  { field: 'Predecessor', headerText: 'Depedency', width: '150' },
  { field: 'TaskName', headerText: 'Task Name', width: '150' },
  { field: 'StartDate', headerText: 'Start Date', width: '150' },
  { field: 'Duration', headerText: 'Duration', width: '150' },
  { field: 'Progress', headerText: 'Progress', width: '150' },
],
});
gantChart.appendTo('#Default');

```

INDEX.HTML

```

<html>
  <head>
    <script
src="https://ej2.syncfusion.com/javascript/demos/gantt/default/datasource.js
" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet"/>
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet"/>
    <style> body { touch-action: none; } </style>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
  </head>
  <body>
    <div class="stackblitz-container material3">
      <div class="control-section">
        <div class="content-wrapper">
          <label>Show/Hide Dependency Line</label>
          <input type="checkbox" id="switch" />
          <div id="Default"></div>
        </div>
      </div>
    </div>
    <script src="index.js" type="text/javascript"></script>
  </body>
</html>

```

Sorting in ##Platform_Name## Gantt control

Sorting enables you to sort data in the ascending or descending order. To sort a column, click the column header.

To sort multiple columns, press and hold the CTRL key and click the column header. You can clear sorting of any one of the multi-sorted columns by pressing and holding the SHIFT key and clicking the specific column header.

To enable sorting in the Gantt control, set the [allowSorting](#) property to true. Sorting options can be configured through the [sortSettings](#) property.

To sort, inject the [Sort](#) module into the Gantt control.

INDEX.JS

```
ej.gantt.Gantt.Inject(ej.gantt.Sort);
var editingData = [
    {
        TaskID: 1,
        TaskName: 'Project Initiation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        subtasks: [
            { TaskID: 2, TaskName: 'Identify Site location', StartDate:
new Date('04/02/2019'), Duration: 4, Progress: 50 },
            { TaskID: 3, TaskName: 'Perform Soil test', StartDate: new
Date('04/02/2019'), Duration: 4, Progress: 50 },
            { TaskID: 4, TaskName: 'Soil test approval', StartDate: new
Date('04/02/2019'), Duration: 4, Progress: 50 },
        ]
    },
    {
        TaskID: 5,
        TaskName: 'Project Estimation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        subtasks: [
            { TaskID: 6, TaskName: 'Develop floor plan for estimation',
StartDate: new Date('04/04/2019'), Duration: 3, Progress: 50 },
            { TaskID: 7, TaskName: 'List materials', StartDate: new
Date('04/04/2019'), Duration: 3, Progress: 50 },
            { TaskID: 8, TaskName: 'Estimation approval', StartDate: new
Date('04/04/2019'), Duration: 3, Progress: 50 }
        ]
    },
];
var ganttChart = new ej.gantt.Gantt({
    dataSource: editingData,
    height: '450px',
    columns: [
        { field: 'TaskID', headerText: 'Task ID', textAlign: 'Left',
width: '100' },
        { field: 'TaskName', headerText: 'Task Name', width: '250' },
        { field: 'StartDate', headerText: 'Start Date', width: '150' },
        { field: 'Duration', headerText: 'Duration', width: '150' },
        { field: 'Progress', headerText: 'Progress', width: '150' },
    ],
    allowSorting: true,
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    }
});
ganttChart.appendTo('#Gantt');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

* Gantt columns are sorted in the ascending order. If you click the already sorted column, the sort direction toggles.

* To disable sorting for a particular column, set the [columns.allowSorting](#) property to false.

Sorting column on Gantt initialization

The Gantt control can be rendered with sorted columns initially, and this can be achieved by using the [sortSettings](#) property. You can add columns that are sorted initially in the [sortSettings.columns](#) collection defined with [field](#) and [direction](#) properties. The following code example shows how to add the sorted column to Gantt initialization.

INDEX.JS

```

ej.gantt.Gantt.Inject(ej.gantt.Sort);
var ganttChart = new ej.gantt.Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',

```



```

        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    sortSettings: { columns: [{ field: 'TaskID', direction:
'Ascending' }, { field: 'TaskName', direction: 'Ascending' }] },
    allowSorting: true
});
gantChart.appendTo('#Gantt');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Sorting column dynamically

Columns in the Gantt control can be sorted dynamically using the [sortColumn](#) method. The following code example demonstrates how to invoke the [sortColumn](#) method by clicking the custom button.

INDEX.JS

```

ej.gantt.Gantt.Inject(ej.gantt.Sort);
var ganttChart = new ej.gantt.Gantt({
  dataSource: GanttData,
  height: '450px',
```

```

        allowSorting: true,
        taskFields: {
            id: 'TaskID',
            name: 'TaskName',
            startDate: 'StartDate',
            duration: 'Duration',
            progress: 'Progress',
            child: 'subtasks'
        }
    });
    ganttChart.appendTo('#Gantt');
    var sortBtn= new ej.buttons.Button();
    sortBtn.appendTo('#sortColumn');
    document.getElementById('sortColumn').addEventListener('click', () => {
        var ganttObj= document.getElementById('Gantt').ej2_instances[0];
        ganttObj.sortModule.sortColumn('TaskName', "Descending", false)
    });

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <button id="sortColumn">Sort TaskName Column</button>

    <div id="container">
        <div id="Gantt"></div>
    </div>
    <script>
    var ele = document.getElementById('container');
    if(ele) {
        ele.style.visibility = "visible";
    }
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

Clear all the sorted columns dynamically

In the Gantt control, you can clear all the sorted columns and return to previous position using the [clearSorting](#) public method. The following code snippet shows how to clear all the sorted columns by clicking the custom button.

INDEX.JS

```
ej.gantt.Gantt.Inject(ej.gantt.Sort);
var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    sortSettings: { columns: [{ field: 'TaskID', direction:
'Ascending' }, { field: 'TaskName', direction: 'Ascending' }] },
    allowSorting: true
});
ganttChart.appendTo('#Gantt');
var clrBtn= new ej.buttons.Button();
clrBtn.appendTo('#clearSorting');
document.getElementById('clearSorting').addEventListener('click', () => {
    ganttChart.clearSorting();
});
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <button id="clearSorting">Clear Sorting</button>

    <div id="container">
        <div id="Gantt"></div>
    </div>
</body>
```

```

var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Sorting events

During the sort action, the Gantt control triggers two events. The [actionBegin](#) event triggers before the sort action starts, and the [actionComplete](#) event triggers after the sort action is completed.

INDEX.JS

```

ej.gantt.Gantt.Inject(ej.gantt.Sort);
var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    allowSorting: true,
    actionBegin: actionHandler,
    actionComplete: actionHandler
});
ganttChart.appendTo('#Gantt');
function actionHandler(args) {
    alert(args.requestType + ' ' + args.type); //custom Action
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```

```

    <div id="container">
      <div id="Gantt"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The `args.requestType` is the current action name. For example, for sorting the `args.requestType`, value is **sorting**.

Sorting Custom Columns

In Gantt, you can sort custom columns of different types like string, numeric, etc., By adding the custom column in the column collection,

you can perform initial sort using the `sortSettings` or you can also sort the column dynamically by a button click.

The following code snippets explains how to achieve this.

INDEX.JS

```

var GanttData = [
  {
    TaskID: 1,
    TaskName: 'Project Initiation',
    StartDate: new Date('04/02/2019'),
    EndDate: new Date('04/21/2019'),
    subtasks: [
      { TaskID: 2, TaskName: 'Identify Site location', StartDate:
new Date('04/02/2019'), Duration: 4, Progress: 50, /*CustomColumn:
'BCustomColumn'*/ CustomColumn: '2' },
      { TaskID: 3, TaskName: 'Perform Soil test', StartDate: new
Date('04/02/2019'), Duration: 4, Progress: 50, /*CustomColumn:
'BCustomColumn'*/ CustomColumn: '3' },
      { TaskID: 4, TaskName: 'Soil test approval', StartDate: new
Date('04/02/2019'), Duration: 4, Progress: 50, /*CustomColumn:
'BCustomColumn'*/ CustomColumn: '4' },
    ]
  },
  {
    TaskID: 5,
    TaskName: 'Project Estimation',
    StartDate: new Date('04/02/2019'),
    EndDate: new Date('04/21/2019'),
    subtasks: [
      { TaskID: 6, TaskName: 'Develop floor plan for estimation',
StartDate: new Date('04/04/2019'), Duration: 3, Progress: 50,
/*CustomColumn: 'BCustomColumn'*/ CustomColumn: '6' },
    ]
  }
]

```

```

        { TaskID: 7, TaskName: 'List materials', StartDate: new
Date('04/04/2019'), Duration: 3, Progress: 50, /*CustomColumn:
'BCustomColumn'*/ CustomColumn: '1' },
        { TaskID: 8, TaskName: 'Estimation approval', StartDate: new
Date('04/04/2019'), Duration: 3, Progress: 50, /*CustomColumn:
'BCustomColumn'*/ CustomColumn: '5' }
    ]
},
];
var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    allowSorting: true,
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    columns: [
        { field: 'TaskID', headerText: 'Task ID' },
        { field: 'Progress', headerText: 'Progress' },
        { field: 'TaskName', headerText: 'Task Name' },
        { field: 'StartDate', headerText: 'Start Date' },
        { field: 'Duration', headerText: 'Duration' },
        { field: 'CustomColumn', headerText: 'CustomColumn' }
    ]
});
ganttChart.appendTo('#Gantt');
var sortBtn= new ej.buttons.Button();
sortBtn.appendTo('#sortColumn');
document.getElementById('sortColumn').addEventListener('click', () => {
    var ganttObj= document.getElementById('Gantt').ej2_instances[0];
    ganttObj.sortModule.sortColumn('CustomColumn', "Ascending", false)
});

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>

```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <button id="sortColumn">Sort Custom Column</button>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Baseline in ##Platform_Name## Gantt control

The baseline feature enables users to view the deviation between the planned dates and actual dates of the tasks in a project. Baseline dates or planned dates of a task may or may not be same as the actual task dates. The baseline can be enabled by setting the [renderBaseline](#) property to true and the baseline color can be changed using the [baselineColor](#) property. To render the baseline, you should map the baseline start and end date values from the data source. This can be done using the [baselineStartDate](#) and [baselineEndDate](#) properties. The following code example shows how to enable a baseline in the Gantt control.

INDEX.JS

```

var GanttData = [
    {
        TaskID: 1,
        TaskName: 'Project Initiation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        subtasks: [
            { TaskID: 2, TaskName: 'Identify Site
location',BaselineStartDate: new Date('04/02/2019'),BaselineEndDate: new
Date('04/06/2019'), StartDate: new Date('04/02/2019'), Duration: 0,
Progress: 50 },
            { TaskID: 3, TaskName: 'Perform Soil
test',BaselineStartDate: new Date('04/04/2019'),BaselineEndDate: new
Date('04/09/2019'), StartDate: new Date('04/02/2019'), Duration: 4,
Progress: 50 },
            { TaskID: 4, TaskName: 'Soil test
approval',BaselineStartDate: new Date('04/08/2019'),BaselineEndDate: new
Date('04/12/2019'), StartDate: new Date('04/02/2019'), Duration: 4,
Progress: 50 },
        ]
    },
    {
        TaskID: 5,
        TaskName: 'Project Estimation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
    }
]

```

```

        subtasks: [
            { TaskID: 6, TaskName: 'Develop floor plan for
estimation',BaselineStartDate: new Date('04/04/2019'),BaselineEndDate: new
Date('04/08/2019'), StartDate: new Date('04/04/2019'), Duration: 3,
Progress: 50 },
            { TaskID: 7, TaskName: 'List materials',BaselineStartDate:
new Date('04/02/2019'),BaselineEndDate: new Date('04/04/2019'), StartDate:
new Date('04/04/2019'), Duration: 3, Progress: 50 },
            { TaskID: 8, TaskName: 'Estimation
approval',BaselineStartDate: new Date('04/02/2019'),BaselineEndDate: new
Date('04/02/2019'), StartDate: new Date('04/04/2019'), Duration: 0,
Progress: 50 }
        ]
    },
    ];
var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        baselineStartDate: "BaselineStartDate",
        baselineEndDate: "BaselineEndDate",
        child: 'subtasks'
    },
    renderBaseline: true,
    baselineColor: 'red'
});
ganttChart.appendTo('#Gantt');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
```



```

        <div id="Gantt"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Timeline

Timeline in ##Platform_Name## Gantt control

In the Gantt control, timeline is used to represent the project duration as individual cells with defined unit and formats.

Timeline view modes

Gantt contains the following in-built timeline view modes:

- Hour
- Day
- Week
- Month
- Year

Timescale mode in Gantt can be defined by using [timelineViewMode](#) property and also we can define timescale mode of top tier and bottom tier by using [topTier.unit](#) and [bottomTier.unit](#) properties.

Week timeline mode

In the **Week** timeline mode, the upper part of the schedule header displays the weeks, whereas the bottom half of the header displays the days. Refer to the following code example.

INDEX.JS

```

var ganttChart = new ej.gantt.Gantt({
    dataSource: WeekData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        dependency: 'Predecessor',
        child: 'subtasks'
    },
    timelineSettings: {
        timelineViewMode: 'Week'
    }
});
ganttChart.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Month timeline mode

In the **Month** timeline mode, the upper part of the schedule header displays the months, whereas the bottom header of the schedule displays its corresponding weeks. Refer to the following code example.

INDEX.JS

```

var ganttChart = new ej.gantt.Gantt({
  dataSource: MonthData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  timelineSettings: {
    timelineViewMode: 'Month',
    timelineUnitSize: 150
  }
});
ganttChart.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Year timeline mode

In the **Year** timeline mode, the upper schedule header displays the years whereas, the bottom header displays its corresponding months. Refer to the following code example.

INDEX.JS

```

var ganttChart = new ej.gantt.Gantt({
  dataSource: YearData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  timelineSettings: {
    timelineViewMode: 'Year',
    timelineUnitSize: 70
  }
});

```

```
ganttChart.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Day timeline mode

In the **Day** timeline mode, the upper part of the header displays the days whereas, the bottom schedule header displays its corresponding hours. Refer to the following code example.

INDEX.JS

```
var ganttChart = new ej.gantt.Gantt({
  dataSource: DayData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  timelineSettings: {
    timelineViewMode: 'Day'
```

```

    }
});
gantChart.appendTo('#Gantt');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Hour timeline mode

An **Hour** timeline mode tracks the tasks in minutes scale. In this mode, the upper schedule header displays hour scale and the lower schedule header displays its corresponding minutes.

INDEX.JS

```

var ganttChart = new ej.gantt.Gantt({
  dataSource: HourData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
});
```

```

        timelineSettings: {
            timelineViewMode: 'Hour'
        }
    });
    ganttChart.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Week start day customization

In the Gantt control, you can customize the week start day using the [weekStartDay](#) property. By default, the [weekStartDay](#) is set to 0, which specifies the Sunday as a start day of the week. But, you can customize the week start day by using the following code example.

INDEX.JS

```

var ganttChart = new ej.gantt.Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',

```

```

        progress: 'Progress',
        child: 'subtasks'
    },
    timelineSettings: {
        timelineViewMode: 'Week',
        weekStartDay: 3
    }
});
gantChart.appendTo('#Gantt');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Customize automatic timescale update action

In the Gantt control, the schedule timeline will be automatically updated when the tasks date values are updated beyond the project start date and end date ranges. This can be enabled or disabled using the [updateTimescaleView](#) property.

INDEX.JS

```

var ganttChart = new ej.gantt.Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
```

```

        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
            duration: 'Duration',
        progress: 'Progress',
            child: 'subtasks'
    },
    timelineSettings: {
        updateTimescaleView: false
    },
    editSettings: {
        allowEditing: true,
        allowTaskbarEditing: true
    }
});
gantChart.appendTo('#Gantt');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Top tier and bottom tier in ##Platform_Name## Gantt control

Gantt control contains two tier layout in timeline, we can customize the top tier and bottom tier using [topTier](#) and [bottomTier](#) properties. Timeline tier's unit can be defined by using [unit](#) property and [format](#) property was used to define date format of timeline cell and [count](#) property was used to define how

many unit will be combined as single cell and [formatter](#) property was used to define custom method to format the date value of timeline cell.

INDEX.JS

```
var ganttChart = new ej.gantt.Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  timelineSettings: {
    topTier: {
      format: 'MMM',
      unit: 'Month'
    },
    bottomTier: {
      unit: 'Day'
    }
  }
});
ganttChart.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
```

```

    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Combining timeline cells

In Gantt, timeline cells in top tier and bottom tier can be combined with number of timeline units, this can be achieved by using [topTier.count](#) and [bottomTier.count](#) properties. Please refer the below sample.

INDEX.JS

```

var ganttChart = new ej.gantt.Gantt({
    dataSource: Data,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    timelineSettings: {
        timelineUnitSize: 100,
        topTier: {
            unit: 'Year'
        },
        bottomTier: {
            unit: 'Month',
            count: 6
        }
    }
});
ganttChart.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>

```

```

<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Format value of timeline cell

In the Gantt control, you can format the value of top and bottom timeline cells using the standard date format string or the custom formatter method. This can be done using the [topTier.format](#), [topTier.formatter](#), [bottomTier.format](#) and [bottomTier.formatter](#) properties. The following example shows how to use the formatter method for timeline cells.

INDEX.JS

```

var ganttChart = new ej.gantt.Gantt({
    dataSource: Data,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    timelineSettings: {
        topTier: {
            unit: 'Month',
            count: 3,
            formatter: (date) => {
                var month = date.getMonth();
                if (month >= 0 && month <=2) {
                    return 'Q1';
                } else if (month >= 3 && month <=5) {
                    return 'Q2';
                } else if (month >= 6 && month <=8) {
                    return 'Q3';
                } else {
                    return 'Q4';
                }
            }
        },
        bottomTier: {
            unit: 'Month',
            format: 'MMM'
        }
    }
},

```

```

        projectStartDate: new Date('01/04/2019'),
        projectEndDate: new Date('12/30/2019')
    });
    ganttChart.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Timeline cell width

In the Gantt control, you can define the width value of timeline cell using the [timelineSettings.timelineUnitSize](#) property. This value will be set to the bottom timeline cell, and the width value of top timeline cell will be calculated automatically based on bottom tier cell width using the [topTier.unit](#) and [timelineSettings.timelineUnitSize](#) properties. Refer to the following example.

INDEX.JS

```

var ganttChart = new ej.gantt.Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',

```

```

        progress: 'Progress',
        child: 'subtasks'
    },
    timelineSettings: {
        timelineUnitSize: 200
    }
});
gantChart.appendTo('#Gantt');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Zooming in ##Platform_Name## Gantt control

The zooming support provides options to increase or decrease the width of timeline cells and also provides options to change the timeline units dynamically. This support enables you to view the tasks in a project clearly from minute to decade timespan. To enable the zooming features, define the **ZoomIn**, **ZoomOut**, and **ZoomToFit** items to toolbar items collections, and this action can be performed on external actions such as button click using the [zoomIn](#), [zoomOut](#), and [fitToProject](#) built-in methods. The following zooming options are available to view the project:

Zoom in

This support is used to increase the timeline width and timeline unit from years to minutes timespan. When the **ZoomIn** icon was clicked, the timeline cell width is increased when the cell size exceeds the specified range and the timeline unit is changed based on the current zoom levels.

Zoom out

This support is used to increase the timeline width and timeline unit from minutes to years timespan. When the **ZoomOut** icon was clicked, the timeline cell width is decreased when the cell size falls behind the specified range and the timeline view mode is changed based on the current zooming levels.

Zoom to fit

This support is used to view all the tasks available in a project within available area on the chart part of Gantt. When users click the **ZoomToFit** icon, then all the tasks are rendered within the available chart container width.

INDEX.JS

```
var ganttChart = new ej.gantt.Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    dependency: 'Predecessor',
    child: 'subtasks'
  },
  toolbar: ['ZoomIn', 'ZoomOut', 'ZoomToFit'],
  labelSettings: {
    leftLabel: 'TaskName'
  },
  projectStartDate: new Date('03/24/2019'),
  projectEndDate: new Date('04/28/2019')
});
ganttChart.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customizing zooming levels

In Gantt, the zoom in and zoom out actions are performed based on the predefined zooming levels in the `zoomingLevels` property. You can customize the zooming actions by defining the required zooming collection to the `zoomingLevels` property.

INDEX.JS

```

var customZoomingLevels = [{
    topTier: { unit: 'Month', format: 'MMM, yy', count: 1 },
    bottomTier: { unit: 'Week', format: 'dd', count: 1 },
    timelineUnitSize: 33, level: 0,
    timelineViewMode: 'Month', weekStartDay: 0,
    updateTimescaleView: true, weekendBackground: null, showTooltip: true
},
{
    topTier: { unit: 'Month', format: 'MMM, yyyy', count: 1 },
    bottomTier: { unit: 'Week', format: 'dd MMM', count: 1 },
    timelineUnitSize: 66, level: 1,
    timelineViewMode: 'Month', weekStartDay: 0,
    updateTimescaleView: true, weekendBackground: null, showTooltip: true
},
{
    topTier: { unit: 'Month', format: 'MMM, yyyy', count: 1 },
    bottomTier: { unit: 'Week', format: 'dd MMM', count: 1 },
    timelineUnitSize: 99, level: 2,
    timelineViewMode: 'Month', weekStartDay: 0,
    updateTimescaleView: true, weekendBackground: null, showTooltip: true
},
{
    topTier: { unit: 'Week', format: 'MMM dd, yyyy', count: 1 },
    bottomTier: { unit: 'Day', format: 'd', count: 1 },
    timelineUnitSize: 33, level: 3,
    timelineViewMode: 'Week', weekStartDay: 0,
    updateTimescaleView: true, weekendBackground: null, showTooltip: true
},
{
    topTier: { unit: 'Week', format: 'MMM dd, yyyy', count: 1 },

```

```

        bottomTier: { unit: 'Day', format: 'd', count: 1 },
        timelineUnitSize: 66, level: 4,
        timelineViewMode: 'Week', weekStartDay: 0,
        updateTimescaleView: true, weekendBackground: null, showTooltip: true
    },
    {
        topTier: { unit: 'Day', format: 'E dd yyyy', count: 1 },
        bottomTier: { unit: 'Hour', format: 'hh a', count: 12 },
        timelineUnitSize: 66, level: 5,
        timelineViewMode: 'Day', weekStartDay: 0,
        updateTimescaleView: true, weekendBackground: null, showTooltip: true
    },
    {
        topTier: { unit: 'Day', format: 'E dd yyyy', count: 1 },
        bottomTier: { unit: 'Hour', format: 'hh a', count: 6 },
        timelineUnitSize: 99, level: 6,
        timelineViewMode: 'Day', weekStartDay: 0,
        updateTimescaleView: true, weekendBackground: null, showTooltip: true
    },
];
var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        dependency: 'Predecessor',
        child: 'subtasks'
    },
    toolbar: ['ZoomIn', 'ZoomOut', 'ZoomToFit'],
    labelSettings: {
        leftLabel: 'TaskName'
    },
    dataBound: function() {
        ganttChart.zoomingLevels = customZoomingLevels;
    },
    projectStartDate: new Date('03/24/2019'),
    projectEndDate: new Date('04/28/2019')
});
ganttChart.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

```



```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Zoom action by methods

You can perform the various zoom actions dynamically or on external click action using the following methods:

- Zoom in - [zoomIn](#)
- Zoom out - [zoomOut](#)
- Fit to project - [fitToProject](#)

INDEX.JS

```

var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        dependency: 'Predecessor',
        child: 'subtasks'
    },
    labelSettings: {
        leftLabel: 'TaskName'
    },
    projectStartDate: new Date('03/24/2019'),
    projectEndDate: new Date('04/28/2019')
});
ganttChart.appendTo('#Gantt');
var zoomInBtn= new ej.buttons.Button();
zoomInBtn.appendTo('#zoomIn');
document.getElementById('zoomIn').addEventListener('click', function() {
    ganttChart.zoomIn();

```

```

});
var zoomOutBtn= new ej.buttons.Button();
zoomInBtn.appendTo('#zoomOut');
document.getElementById('zoomOut').addEventListener('click', function() {
    ganttChart.zoomOut();
});
var fitToBtn= new ej.buttons.Button();
zoomInBtn.appendTo('#fitToProject');
document.getElementById('fitToProject').addEventListener('click', function() {
    ganttChart.fitToProject();
});

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="zoomIn">ZoomIn</button>
        <button id="zoomOut">ZoomOut</button>
        <button id="fitToProject">FitToProject</button>
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Timezone in ##Platform_Name## Gantt control

The Gantt uses the current system time zone by default. If it needs to follow some other user-specified time zone, then the `timezone` property must be used.

Understanding date manipulation in JavaScript

The `new Date()` in JavaScript returns the exact current date object with complete time and timezone information. For example, it may return a value such as `Wed Dec 12, 2018, 05:23:27 GMT+0530 (India Standard Time)` which indicates that the current date is December 12, 2018, and the current time is 5.23 AM on browsers following the IST timezone.

Display same time everywhere with no time difference

Setting `timezone` to UTC for Gantt will display the same time as in the database for all the users in different time zone.

INDEX.JS

```
var ganttChart = new ej.gantt.Gantt({
    dataSource: [
        { taskID: 1, taskName: 'Project Schedule', startDate:
new Date('02/04/2019 08:00'), endDate: new Date('03/10/2019') },
        { taskID: 2, taskName: 'Planning', startDate: new
Date('02/04/2019 08:00'), endDate: new Date('02/10/2019'), parentID: 1 },
        { taskID: 3, taskName: 'Plan timeline', startDate: new
Date('02/04/2019 08:00'), endDate: new Date('02/10/2019'), duration: 6,
progress: '60', parentID: 2 },
        { taskID: 4, taskName: 'Plan budget', startDate: new
Date('02/04/2019 08:00'), endDate: new Date('02/10/2019'), duration: 6,
progress: '90', parentID: 2 },
        { taskID: 5, taskName: 'Allocate resources', startDate: new
Date('02/04/2019 08:00'), endDate: new Date('02/10/2019'), duration: 6,
progress: '75', parentID: 2 },
        { taskID: 6, taskName: 'Planning complete', startDate: new
Date('02/06/2019 08:00'), endDate: new Date('02/10/2019'), duration: 0,
predecessor: '3FS,4FS,5FS', parentID: 2 },
        { taskID: 7, taskName: 'Design', startDate: new Date('02/13/2019
08:00'), endDate: new Date('02/17/2019 08:00'), parentID: 1 },
        { taskID: 8, taskName: 'Software Specification', startDate: new
Date('02/13/2019 08:00'), endDate: new Date('02/15/2019'), duration: 3,
progress: '60', predecessor: '6FS', parentID: 7 },
        { taskID: 9, taskName: 'Develop prototype', startDate: new
Date('02/13/2019 08:00'), endDate: new Date('02/15/2019'), duration: 3,
progress: '100', predecessor: '6FS', parentID: 7 },
        { taskID: 10, taskName: 'Get approval from customer', startDate:
new Date('02/16/2019 08:00'), endDate: new Date('02/17/2019 08:00'),
duration: 2, progress: '100', predecessor: '9FS', parentID: 7 },
        { taskID: 11, taskName: 'Design complete', startDate: new
Date('02/17/2019 08:00'), endDate: new Date('02/17/2019 08:00'), duration:
0, predecessor: '10FS', parentID: 7 }
    ],
    taskFields: {
        id: 'taskID',
        name: 'taskName',
        startDate: 'startDate',
        duration: 'duration',
        progress: 'progress',
        dependency: 'predecessor',
        parentID: 'parentID'
    },
    timelineSettings: {
        timelineUnitSize: 65,
```

```

        topTier: {
            unit: 'Day',
            format: 'MMM dd, yyyy'
        },
        bottomTier: {
            unit: 'Hour',
            format: 'hh:mm a'
        }
    },
    timezone: 'UTC',
    durationUnit: 'Hour',
    includeWeekend: true,
    dateFormat: 'hh:mm a',
    height: '450px',
    dayWorkingTime: [{ from: 0, to: 23 }],
});
gantChart.appendTo('#Gantt');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>
```

CRUD operations with timezone

CRUD operations can be performed with timezone and the Gantt is rendered based on the timezone specified in the load time. All the editing actions will be done based on the user timezone, but on

database save action, we have reversed this conversion to local time and provided data to client-side events for better understanding. Refer to the following code example.

INDEX.JS

```
var gantt = new ej.gantt.Gantt({
  dataSource: [
    {
      taskID: 1,
      taskName: 'Project Schedule',
      startDate: new Date('02/04/2019 08:00'),
      endDate: new Date('03/10/2019')
    },
    {
      taskID: 2,
      taskName: 'Planning',
      startDate: new Date('02/04/2019 08:00'),
      endDate: new Date('02/10/2019'),
      parentID: 1
    },
    {
      taskID: 3,
      taskName: 'Plan timeline',
      startDate: new Date('02/04/2019 08:00'),
      endDate: new Date('02/10/2019'),
      duration: 6,
      progress: '60',
      parentID: 2
    },
    {
      taskID: 4,
      taskName: 'Plan budget',
      startDate: new Date('02/04/2019 08:00'),
      endDate: new Date('02/10/2019'),
      duration: 6,
      progress: '90',
      parentID: 2
    },
    {
      taskID: 5,
      taskName: 'Allocate resources',
      startDate: new Date('02/04/2019 08:00'),
      endDate: new Date('02/10/2019'),
      duration: 6,
      progress: '75',
      parentID: 2
    },
    {
      taskID: 6,
      taskName: 'Planning complete',
      startDate: new Date('02/06/2019 08:00'),
      endDate: new Date('02/10/2019'),
      duration: 0,
      predecessor: '3FS,4FS,5FS',
      parentID: 2
    }
  ]
});
```

```

        taskID: 7,
        taskName: 'Design',
        startDate: new Date('02/13/2019 08:00'),
        endDate: new Date('02/17/2019 08:00'),
        parentID: 1
    },
    {
        taskID: 8,
        taskName: 'Software Specification',
        startDate: new Date('02/13/2019 08:00'),
        endDate: new Date('02/15/2019'),
        duration: 3,
        progress: '60',
        predecessor: '6FS',
        parentID: 7
    },
    {
        taskID: 9,
        taskName: 'Develop prototype',
        startDate: new Date('02/13/2019 08:00'),
        endDate: new Date('02/15/2019'),
        duration: 3,
        progress: '100',
        predecessor: '6FS',
        parentID: 7
    },
    {
        taskID: 10,
        taskName: 'Get approval from customer',
        startDate: new Date('02/16/2019 08:00'),
        endDate: new Date('02/17/2019 08:00'),
        duration: 2,
        progress: '100',
        predecessor: '9FS',
        parentID: 7
    },
    {
        taskID: 11,
        taskName: 'Design complete',
        startDate: new Date('02/17/2019 08:00'),
        endDate: new Date('02/17/2019 08:00'),
        duration: 0,
        predecessor: '10FS',
        parentID: 7
    }
],
editSettings: {
    allowEditing: true,
    allowDeleting: true,
    allowTaskbarEditing: true,
    showDeleteConfirmDialog: true
},
taskFields: {
    id: 'taskID',
    name: 'taskName',
    startDate: 'startDate',
    duration: 'duration',

```

```

        progress: 'progress',
        dependency: 'predecessor',
        parentID: 'parentID'
    },
    timelineSettings: {
        timelineUnitSize: 65,
        topTier: {
            unit: 'Day',
            format: 'MMM dd, yyyy'
        },
        bottomTier: {
            unit: 'Hour',
            format: 'hh:mm a'
        }
    },
    timezone: 'America/New_York',
    durationUnit: 'Hour',
    includeWeekend: true,
    dateFormat: 'hh:mm a',
    height: '450px',
    dayWorkingTime: [{ from: 0, to: 23 }]
    });
function actionComplete(args) {
    if (args.action == "TaskbarEditing") {
        console.log(args.data.ganttProperties.endDate);
    }
}
gantt.appendTo('#Gantt');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
```

```

    ele.style.visibility = "visible";
  }
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Timezone methods

offset

This method is used to calculate the difference between the passed UTC date and timezone.

Parameters	Type	Description
----- ----- -----		
Date	Date	UTC time as date object.
Timezone	String	Timezone.

Returns **number**

```

`ts
// Assume your local timezone as IST/UTC+05:30
let timezone: Timezone = new Timezone();
let date: Date = new Date(2018,11,5,15,25,11);
let timeZoneOffset: number = timezone.offset(date,"Europe/Paris");
console.log(timeZoneOffset); //-60
`

```

convert

This method is used to convert the passed date from one timezone to another.

Parameters	Type	Description
----- ----- -----		
Date	Date	UTC time as date object.
fromOffset	number/string	Timezone from which date needs to be converted.
toOffset	number/string	Timezone to which date needs to be converted.

Returns **Date**

```

`ts
// Assume your local timezone as IST/UTC+05:30
let timezone: Timezone = new Timezone();
let date: Date = new Date(2018,11,5,15,25,11);
let convertedDate: Date = timezone.convert(date, "Europe/Paris", "Asia/Tokya");
let convertedDate1: Date = timezone.convert(date, 60, -360);
console.log(convertedDate); //2018-12-05T08:55:11.000Z
`

```



```
console.log(convertedDate1); //2018-12-05T16:55:11.000Z
```

```
,
```

[remove](#)

This method is used to remove the time difference between passed UTC date and timezone.

Parameters	Type	Description
----- ----- -----		
Date	Date	UTC as date object.
Timezone	String	Timezone.

Returns **Date**

```
`ts
```

```
// Assume your local timezone as IST/UTC+05:30
```

```
let timezone: Timezone = new Timezone();
```

```
let date: Date = new Date(2018,11,5,15,25,11);
```

```
let convertedDate: Date = timezone.remove(date, "Europe/Paris");
```

```
console.log(convertedDate); //2018-12-05T14:25:11.000Z
```

```
,
```

[Event markers in ##Platform_Name## Gantt control](#)

The event markers in the Gantt control is used to highlight the important events in a project. Event markers can be initialized by using the [eventMarkers](#) property, and you can define date and label for the event markers using the [day](#) and [label](#) properties. You can also customize it using the [cssClass](#) properties. The following code example shows how to add event markers in the Gantt control.

To highlight the days, inject the [DayMarkers](#) module into the Gantt control.

INDEX.JS

```
ej.gantt.Gantt.Inject(ej.gantt.DayMarkers);
var ganttChart = new ej.gantt.Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  eventMarkers: [
    {
      day: '04/10/2019',
      cssClass: 'e-custom-event-marker',
      label: 'Project approval and kick-off'
    }
  ]
});
```

```
});
ganttChart.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <style>
    .e-gantt .e-gantt-chart .e-custom-event-marker {
      width: 1px;
      border-left: 2px green dotted;
    }
  </style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Displaying eventMarkers in stacked manner

When [eventMarkers](#) are given in consecutive dates and zoomToFit is performed, they may overlap. To avoid this, you can update the position of the eventMarkers in the [dataBound](#) and [actionComplete](#) events so that they are not overlapped and are visible to read.

INDEX.JS

```
ej.gantt.Gantt.Inject(ej.gantt.DayMarkers,ej.gantt.Toolbar);
function updateEventMarker() {
  document.getElementsByClassName('e-span-label')[1].style.top = '100px';
  document.getElementsByClassName('e-gantt-right-arrow')[1].style.top =
    '110px';
}
```

```

}
var ganttChart = new ej.gantt.Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    endDate: 'EndDate',
    duration: 'Duration',
    progress: 'Progress',
    dependency: 'Predecessor',
    child: 'subtasks',
  },
  eventMarkers: [
    {
      day: new Date('04/02/2019'),
    },
    {
      day: new Date('04/09/2019'),
      label: 'Research phase research phase research phase',
    },
    {
      day: new Date('04/10/2019'),
      label: 'Design phase',
    },
    {
      day: new Date('05/23/2019'),
      label: 'Production phase',
    },
    {
      day: new Date('06/20/2019'),
      label: 'Sales and marketing phase',
    },
  ],
  toolbar: ['ZoomIn', 'ZoomOut', 'ZoomToFit'],
  dataBound: function () {
    updateEventMarker();
    ganttChart.fitToProject();
  },
  actionComplete: function () {
    updateEventMarker();
  },
  projectStartDate: new Date('03/24/2019'),
  projectEndDate: new Date('07/06/2019')
});
ganttChart.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">

```

```

<link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <style>
    .e-gantt .e-gantt-chart .e-custom-event-marker {
      width: 1px;
      border-left: 2px green dotted;
    }
  </style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Label positions in ##Platform_Name## gantt control

The EJ2 Gantt chart offers powerful features for customizing various labels position within the chart, enabling users to present relevant project information clearly. In EJ2 Gantt chart, [labelSettings](#) feature provides three key options for label customization: [rightLabel](#), [taskLabel](#), and [leftLabel](#). Label positions can be initialized by using the [labelSettings](#) property.

The following code example shows how to add label positions in the gantt control.

INDEX.JS

```

ej.gantt.Gantt.Inject(ej.gantt.Selection);
var gantt = new ej.gantt.Gantt({
  dataSource: GanttData,
  resources: resourceCollection,
  viewType: 'ResourceView',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    endDate: 'EndDate',
    duration: 'Duration',
    progress: 'Progress',
    resourceInfo: 'resources',
    work: 'work',

```

```

        child: 'subtasks'
    },
    resourceFields: {
        id: 'resourceId',
        name: 'resourceName',
        unit: 'resourceUnit',
        group: 'resourceGroup'
    },
    columns: [
        { field: 'TaskID', visible: false },
        { field: 'TaskName', headerText: 'Name', width: 250 },
        { field: 'work', headerText: 'Work' },
        { field: 'Progress' },
        { field: 'resourceGroup', headerText: 'Group' },
        { field: 'StartDate' },
        { field: 'Duration' },
    ],
    labelSettings: {
        rightLabel: 'resources',
        leftLabel: 'TaskName',
        taskLabel: '${Progress}%'
    },
    splitterSettings: {
        columnIndex: 3
    },
    height: '450px',
    projectStartDate: new Date('03/22/2019'),
    projectEndDate: new Date('05/18/2019')
});
gantt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>

```

```

<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Managing event marker overlapping in ##Platform_Name## gantt control

In the EJ2 Gantt control, it is possible to customize multiple [eventMarkers](#) for the same date. However, by default, in such scenarios, these markers may overlap each other, resulting in visual clutter. To manage this, the following sample code demonstrates how to utilize the Gantt dataBound function to obtain label and arrow classes. It performs a loop action to fulfill the current requirement and to avoid overlapping. For further clarification, the code snippet below illustrates the flow of its implementation.

INDEX.JS

```

ej.gantt.Gantt.Inject(ej.gantt.DayMarkers);
var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    eventMarkers: [
        {
            day: '04/10/2019',
            cssClass: 'e-custom-event-marker',
            label: 'Project approval and kick-off'
        },
        {
            day: '04/10/2019',
            cssClass: 'e-custom-event-marker',
            label: 'Project approval and kick-off'
        }
    ],
    dataBound() {
        var labeltop = 100;
        var rightarrow = 110;
        for (var i = 0; i < this.eventMarkers.length; i++) {
            document.getElementsByClassName('e-span-label')[i].style.top =
            labeltop + 'px';
            document.getElementsByClassName('e-gantt-right-
            arrow')[i].style.top = rightarrow + 'px';
            labeltop = labeltop + 35;
            rightarrow = rightarrow + 35;
        }
    },
    projectStartDate: new Date('03/24/2019'),

```

```

        projectEndDate: new Date('07/06/2019')
    });
    ganttChart.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <style>
        .e-gantt .e-gantt-chart .e-custom-event-marker {
            width: 1px;
            border-left: 2px green dotted;
        }
    </style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Work in ##Platform_Name## Gantt control

The work is the total hours required to complete a task. Work can be mapped from the data source field using the property [taskFields.work](#). Work can be measured in Hour, Day, Minute. By default, work is measured in Hour and it can be changed, by using the property [workUnit](#).

Note: When the work field is mapped from the data source, the default task type will be FixedWork.

INDEX.JS

```

ej.gantt.Gantt.Inject(ej.gantt.Edit,ej.gantt.Toolbar,ej.gantt.Selection);
var gantt = new ej.gantt.Gantt({

```

```

dataSource: GanttData,
taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    resourceInfo: 'resources',
    work: 'work',
    child: 'subtasks'
},
editSettings: {
    allowAdding: true,
    allowEditing: true,
    allowDeleting: true,
    allowTaskbarEditing: true,
    showDeleteConfirmDialog: true
},
resources: resourceResources,
resourceFields: {
    id: 'resourceId',
    name: 'resourceName',
    unit: 'unit'
},
workUnit: 'Hour',
toolbar: ['Add', 'Edit', 'Update', 'Delete', 'Cancel', 'ExpandAll',
'CollapseAll'],
allowSelection: true,
height: '450px',
treeColumnIndex: 1,
columns: [
    { field: 'TaskID', visible: false },
    { field: 'TaskName', headerText: 'Task Name', width: '180' },
    { field: 'resources', headerText: 'Resources', width: '160' },
    { field: 'work', width: '110' },
    { field: 'Duration', width: '100' },
],
});
gantt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>

```



```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>

<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Task type

The work, duration and resource unit fields of a task depends upon each other and will change automatically on editing any one of these fields. But we can also set these field's values as constant using the [taskType](#) property. **FixedUnit** is the default [taskType](#). The following values can be set to the [taskType](#) property,

- **FixedDuration** - Duration task field will remain constant while updating resource unit or work field.
- **FixedWork** - Work field will remain constant while updating resource unit or duration fields.
- **FixedUnit** - Resource units will remain constant while updating duration or work field.

INDEX.JS

```

ej.gantt.Gantt.Inject(ej.gantt.Edit,ej.gantt.Toolbar,ej.gantt.Selection);
var gantt = new ej.gantt.Gantt({
    dataSource: GanttData,
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        resourceInfo: 'resources',
        work: 'Work',
        child: 'subtasks'
    },
    editSettings: {
        allowAdding: true,
        allowEditing: true,
        allowDeleting: true,
        allowTaskbarEditing: true,
        showDeleteConfirmDialog: true
    },
    taskType: 'FixedWork',
    resources: resourceResources,

```

```

resourceFields: {
    id: 'resourceId',
    name: 'resourceName',
    unit: 'Unit'
},
workUnit: 'Hour',
toolbar: ['Add', 'Edit', 'Update', 'Delete', 'Cancel', 'ExpandAll',
'CollapseAll'],
allowSelection: true,
height: '450px',
treeColumnIndex: 1,
columns: [
    { field: 'TaskID', visible: false },
    { field: 'TaskName', headerText: 'Task Name', width: '180' },
    { field: 'resources', headerText: 'Resources', width: '160' },
    { field: 'work', width: '110' },
    { field: 'Duration', width: '100' },
    { field: 'taskType', headerText: 'Task Type', width: '110' }
],
});
ganttt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>

  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Following table explains how the work, duration and resource unit fields will gets updated on changing any of the fields

Task Type | Changes in Duration | Changes in work | Changes in Resource Units

Fixed Duration | Work field updates | Resource unit updates | Work field updates

Fixed Work | Resource unit updates. Note: For manually scheduled task work will update. | Duration field updates. Note: For manually scheduled task resource unit updates. | Duration will update. Note: For manually scheduled task work field updates.

Fixed Unit | Work field updates | Duration field updates. Note: For manually scheduled task resource unit updates. | Duration will update. Note: For manually scheduled task work field updates.

Note: 1. Fixed Unit is the default taskType in Gantt. 2. The above calculations are not applicable for Milestones.

Resources in ##Platform_Name## Gantt control

In Gantt, the resources are represented by staff, equipment and materials etc. In Gantt control you can show or allocate the resources (human resources) for each task.

Resource collection

The resource collection contains details about resources that are used in the project. Resources are JSON object that contains id, name, unit and group of the resources and this collection is mapped to the Gantt control using the [resources](#) property. These resource fields are mapped to the Gantt control using the [resourceFields](#) property.

Resource fields | Description

[id](#) | This field is used to assign resources to the tasks.

[name](#) | This field is used to map the resource names. These names are displayed as one of Gantt columns and also can display as labels using the [labelSettings](#) property.

[unit](#) | It indicates the amount of work that can be done by a resource for the task in a day.

[group](#) | This field is used to group the resources and the tasks assigned to that particular resource into category.

The following code snippets shows resource collection and how it assigned to Gantt control.

```
`ts
```

```
let projectResources: object[] = resources: [
  { resourceid: 1, resourceName: 'Martin Tamer', resourceGroup: 'Planning Team', resourceUnit: 50},
  { resourceid: 2, resourceName: 'Rose Fuller', resourceGroup: 'Testing Team', resourceUnit: 70 },
  { resourceid: 3, resourceName: 'Margaret Buchanan', resourceGroup: 'Approval Team' },
  { resourceid: 4, resourceName: 'Fuller King', resourceGroup: 'Development Team' },
  { resourceid: 5, resourceName: 'Davolio Fuller', resourceGroup: 'Approval Team' },
  { resourceid: 6, resourceName: 'Van Jack', resourceGroup: 'Development Team', resourceUnit: 40 },
];
let gantt: Gantt = new Gantt({
```

```
resourceFields: {
id: 'resourceId', //resource Id Mapping
name: 'resourceName', //resource Name mapping
unit: 'resourceUnit', //resource Unit mapping
group: 'resourceGroup' //resource Group mapping
},
resources: projectResources //resource collection dataSource
});
gantt.appendTo('#Gantt');
```

Assign resource

We can assign resources for a task at initial load, using the resource id value of the resources as a collection. This collection is mapped from the dataSource to the Gantt control using the [resourceInfo](#) property.

Resources are assigned to tasks in following ways.

Assign resource alone

If the unit is not specified for specific resource, the amount of work done will be consider as 100% by default. In such cases, the resource unit will not be displayed in Gantt UI.

```
`ts
{ TaskID: 2, TaskName: 'Identify site location', StartDate: new Date('04/02/2019'), Duration: 4, Progress:
50, resources: [1] },
```

Assign resource with unit

We can assign the quantity of work done by the resources for the specific task as like below code snippet.

```
`ts
{ TaskID: 3, TaskName: 'Perform soil test', StartDate: new Date('03/29/2019'), Duration: 4,
resources: [{resourceId: 2, resourceUnit: 70}, {resourceId: 1, resourceUnit: 70}] },
```

When resource unit is defined in resource collection, the amount of work done by that particular resource will be same for all the tasks.

The following code snippet shows how to assign the resource for each task and map to Gantt control.

INDEX.JS

```
var ganttChart = new ej.gantt.Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
```

```

        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        resourceInfo: 'resources',
        child: 'subtasks'
    },
    columns: [
        { field: 'TaskID', visible: false },
        { field: 'TaskName', headerText: 'Task Name', width: '180' },
        { field: 'resources', headerText: 'Resources', width: '160' },
        { field: 'Duration', width: '100' },
    ],
    labelSettings: {
        rightLabel: 'resources'
    },
    height: '450px',
    resourceFields: {
        id: 'resourceId',
        name: 'resourceName',
        unit: 'resourceUnit',
        group: 'resourceGroup'
    },
    resources: ProjectResources,
    splitterSettings: {
        columnIndex: 5.1
    }
});
gantChart.appendTo('#Gantt');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
</script>
```

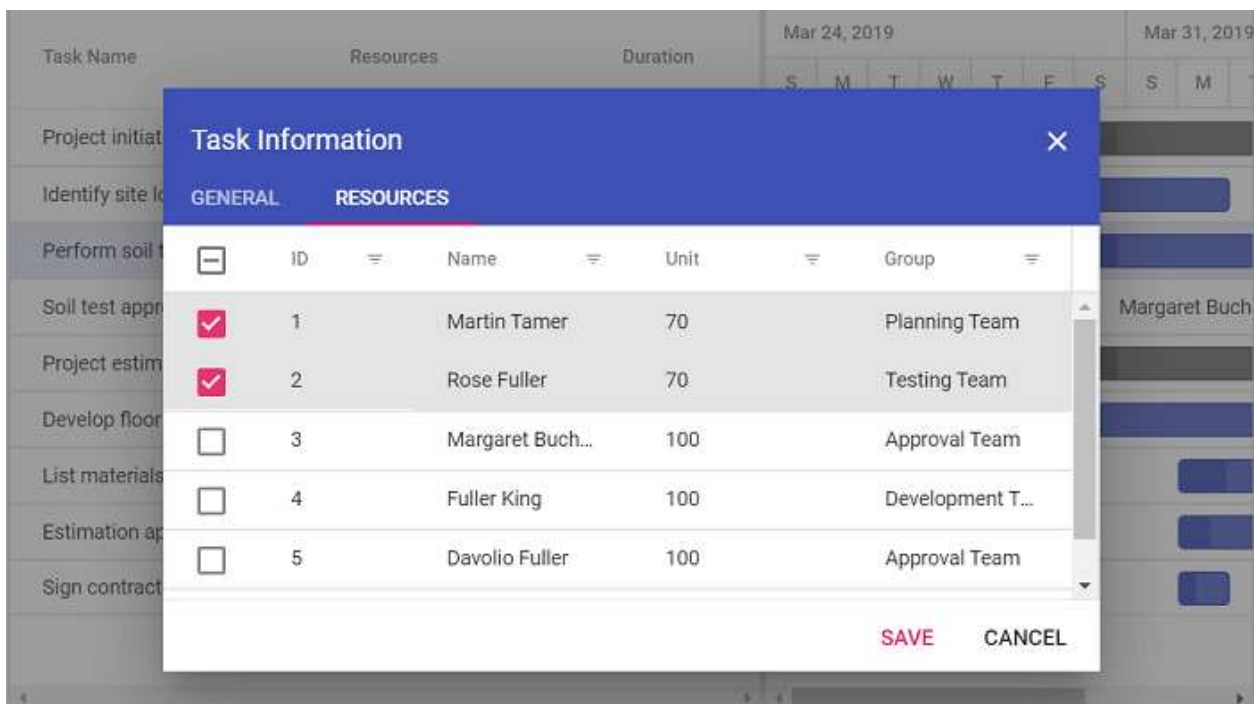
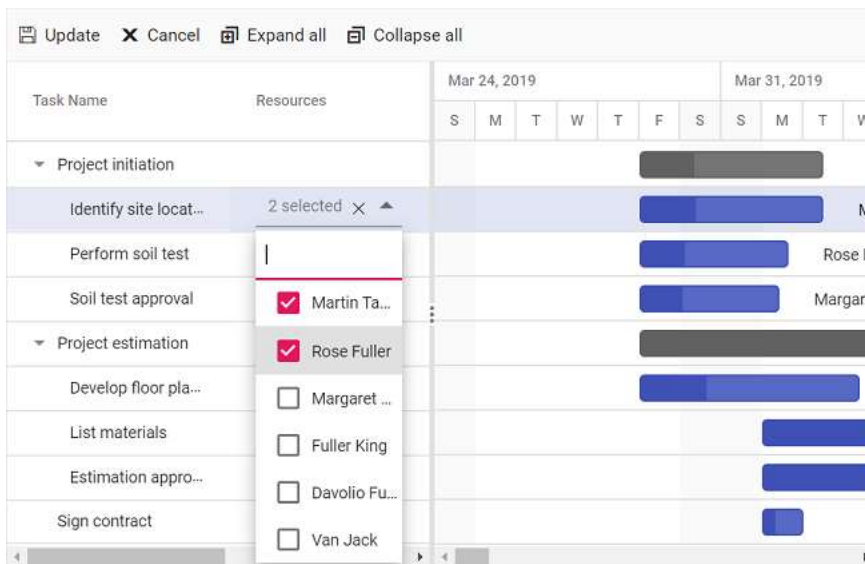
```

var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Add/Edit resource collection

By using cell/ dialog edit option, we can add/remove the multiple resources for a particular task. Resource Unit can be change for a each task on resource tab in the edit dialog by double click on the unit cell.



Resource view in ##Platform_Name## Gantt control

The resource breakdown view is used to visualize the tasks assigned to each resource in hierarchy manner. Resources are displayed as parents and all the tasks assigned to each resource are displayed as its child records. It can be initialized by setting the [viewType](#) property to **ResourceView**.

Unassigned task

A task not assigned to any one of the resource are termed as unassigned tasks. The unassigned tasks are grouped with a name as **Unassigned Task** and displayed at the bottom of Gantt data collection . It is validated at load time during Gantt record creation by default based on a task **resourceInfo** mapping property in the Gantt chart data source. If the resource is assigned to the unassigned grouped tasks, the task will be moved as child to the respective resource.

Resource task

A task assigned to one or more resources are termed as resource task and it is added as child task to the respective resource. Already assigned task can also be shared or moved with other resources by adding a resource name to the task or removing resource name from the task by cell or dialog editing.

Note: Currently there is no support for unscheduled task in Resource view Gantt.

INDEX.JS

```
ej.gantt.Gantt.Inject(ej.gantt.Edit,ej.gantt.Toolbar,ej.gantt.Selection);
var gantt = new ej.gantt.Gantt({
  dataSource: GanttData,
  resources: resourceCollection,
  viewType: 'ResourceView',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    endDate: 'EndDate',
    duration: 'Duration',
    progress: 'Progress',
    resourceInfo: 'resources',
    work: 'work',
    child: 'subtasks'
  },
  resourceFields: {
    id: 'resourceId',
    name: 'resourceName',
    unit: 'resourceUnit',
    group: 'resourceGroup'
  },
  editSettings: {
    allowAdding: true,
    allowEditing: true,
    allowDeleting: true,
    allowTaskbarEditing: true,
    showDeleteConfirmDialog: true
  },
  columns: [
    { field: 'TaskID', visible: false },
    { field: 'TaskName', headerText: 'Name', width: 250 },
    { field: 'work', headerText: 'Work' },
    { field: 'Progress' },
  ]
});
```

```

        { field: 'resourceGroup', headerText: 'Group' },
        { field: 'StartDate' },
        { field: 'Duration' },
    ],
    toolbar: ['Add', 'Edit', 'Update', 'Delete', 'Cancel', 'ExpandAll',
'CollapseAll'],
    labelSettings: {
        rightLabel: 'resources'
    },
    splitterSettings: {
        columnIndex: 3
    },
    allowResizing: true,
    allowSelection: true,
    highlightWeekends: true,
    treeColumnIndex: 1,
    height: '450px',
    projectStartDate: new Date('03/28/2019'),
    projectEndDate: new Date('05/18/2019')
});
ganttt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>

    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```


Resource OverAllocation

When a resource is assigned too much of work to complete within a day of resource's available time then it is called as overallocation.

The available working time of resources for completing the task in a day will be calculated based on the `dayWorkingTime` property and `resource unit`.

The range of overallocation dates can be highlighted by a square bracket. It can be enabled by setting the `showOverallocation` property as `true`. The following code example demonstrates how to hide or show the over allocation by clicking the custom button.

Note: By default, the `showOverAllocation` property value is `false`.

INDEX.JS

```
ej.gantt.Gantt.Inject(ej.gantt.Edit,ej.gantt.Toolbar,ej.gantt.Selection);
var gantt = new ej.gantt.Gantt({
  dataSource: overAllocationData,
  resources: resources,
  viewType: 'ResourceView',
  showOverAllocation: true,
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    endDate: 'EndDate',
    duration: 'Duration',
    progress: 'Progress',
    dependency: 'Predecessor',
    resourceInfo: 'resources',
    work: 'work',
    child: 'subtasks'
  },
  resourceFields: {
    id: 'resourceId',
    name: 'resourceName',
    unit: 'resourceUnit',
    group: 'resourceGroup'
  },
  editSettings: {
    allowAdding: true,
    allowEditing: true,
    allowDeleting: true,
    allowTaskbarEditing: true,
    showDeleteConfirmDialog: true
  },
  columns: [
    { field: 'TaskID', visible: false },
    { field: 'TaskName', headerText: 'Name', width: 250 },
    { field: 'work', headerText: 'Work' },
    { field: 'Progress' },
    { field: 'resourceGroup', headerText: 'Group' },
    { field: 'StartDate' },
    { field: 'Duration' },
  ],
  toolbarClick: function (args) {
    if (args.item.id === 'showhidebar') {
```

```

        gantt.showOverAllocation = gantt.showOverAllocation ? false
: true;
    },
    toolbar: ['Add', 'Edit', 'Update', 'Delete', 'Cancel', 'ExpandAll',
'CollapseAll',
    { text: 'Show/Hide Overallocation', tooltipText: 'Show/Hide
Overallocation', id: 'showhidebar' }],
    labelSettings: {
        rightLabel: 'resources',
        taskLabel: 'Progress'
    },
    projectStartDate: new Date('03/28/2019'),
    projectEndDate: new Date('05/18/2019')
});
gantt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>

  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Resource Multi Taskbar

To visualize multiple tasks assigned to each resource in a row when the records are in the collapsed state. It can be enabled by settings the `enableMultiTaskbar` property value as `true`.

The collapse or expand action of a resource record can be achieved only by using the tree grid side arrow icon. Because it will be disabled on chart side action for this support.

When a resource has multiple tasks scheduled on the same date, then the tasks will be overlapped one another. Taskbar editing is also possible to change the task scheduling on the collapsed state.

Note: By default, the `enableMultiTaskbar` property value is `false`.

INDEX.JS

```
ej.gantt.Gantt.Inject(ej.gantt.Edit,ej.gantt.Toolbar,ej.gantt.Selection);
var gantt = new ej.gantt.Gantt({
  dataSource: overAllocationData,
  resources: resources,
  viewType: 'ResourceView',
  showOverAllocation: true,
  enableMultiTaskbar: true,
  collapseAllParentTasks: true,
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    endDate: 'EndDate',
    duration: 'Duration',
    progress: 'Progress',
    dependency: 'Predecessor',
    resourceInfo: 'resources',
    work: 'work',
    expandState: 'isExpand',
    child: 'subtasks'
  },
  resourceFields: {
    id: 'resourceId',
    name: 'resourceName',
    unit: 'resourceUnit',
    group: 'resourceGroup'
  },
  editSettings: {
    allowAdding: true,
    allowEditing: true,
    allowDeleting: true,
    allowTaskbarEditing: true,
    showDeleteConfirmDialog: true
  },
  columns: [
    { field: 'TaskID' },
    { field: 'TaskName', headerText: 'Name', width: 250 },
    { field: 'work', headerText: 'Work' },
    { field: 'Progress' },
    { field: 'resourceGroup', headerText: 'Group' },
    { field: 'StartDate' },
    { field: 'Duration' },
  ],
  toolbar: ['Add', 'Edit', 'Update', 'Delete', 'Cancel', 'ExpandAll', 'CollapseAll'],
  labelSettings: {
    rightLabel: 'resources',
  }
});
```

```

        taskLabel: 'TaskName'
    },
    projectStartDate: new Date('03/28/2019'),
    projectEndDate: new Date('05/18/2019')
});
ganttt.appendTo('#Gantt');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>

  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Enable taskbar drag and drop

In Gantt, you can enable taskbar drag and drop between resources by using the [allowTaskbarDragAndDrop](#) property. This allows you to move a taskbar from one resource to another vertically, making it easier to schedule tasks and manage resources.

INDEX.JS

```

ej.gantt.Gantt.Inject(ej.gantt.Edit,ej.gantt.Toolbar,ej.gantt.Selection);
var gantt = new ej.gantt.Gantt({
  dataSource: overAllocationData,
  resources: resources,
  viewType: 'ResourceView',
  showOverAllocation: true,
  allowTaskbarOverlap: false,
```

```

enableMultiTaskbar: true,
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    endDate: 'EndDate',
    duration: 'Duration',
    progress: 'Progress',
    dependency: 'Predecessor',
    resourceInfo: 'resources',
    work: 'work',
    expandState: 'isExpand',
    child: 'subtasks'
  },
  resourceFields: {
    id: 'resourceId',
    name: 'resourceName',
    unit: 'resourceUnit',
    group: 'resourceGroup'
  },
  editSettings: {
    allowAdding: true,
    allowEditing: true,
    allowDeleting: true,
    allowTaskbarEditing: true,
    showDeleteConfirmDialog: true
  },
  columns: [
    { field: 'TaskID' },
    { field: 'TaskName', headerText: 'Name', width: 250 },
    { field: 'work', headerText: 'Work' },
    { field: 'Progress' },
    { field: 'resourceGroup', headerText: 'Group' },
    { field: 'StartDate' },
    { field: 'Duration' },
  ],
  toolbar: ['ExpandAll', 'CollapseAll'],
  labelSettings: {
    rightLabel: 'resources',
    taskLabel: 'TaskName'
  },
  projectStartDate: new Date('03/28/2019'),
  projectEndDate: new Date('05/18/2019')
});
gantt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>

<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Disable taskbar overlap

In Gantt, you can disable taskbar overlap between resource tasks using the [allowTaskbarOverlap](#) property. This prevents the taskbars for different tasks from overlapping on the same row, making it easier to distinguish between the different tasks and manage resources effectively.

When `allowTaskbarOverlap` is set to false, the resources are displayed in a single row and the row height will be extended to occupy the tasks of the resource when it is in a collapsed state. This view allows you to easily identify any overallocation of tasks for a resource in a project.

It's important to note that when `allowTaskbarOverlap` is disabled, task dependencies or relationships cannot be established between tasks that are rendered in multiple lines for the same resource. If you need to establish dependencies between tasks for the same resource, you may want to consider enabling taskbar overlap.

INDEX.JS

```

ej.gantt.Gantt.Inject(ej.gantt.Edit,ej.gantt.Toolbar,ej.gantt.Selection);
var gantt = new ej.gantt.Gantt({
    dataSource: overAllocationData,
    resources: resources,
    viewType: 'ResourceView',
    showOverAllocation: true,
    allowTaskbarDragAndDrop: true,
    enableMultiTaskbar: true,
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        endDate: 'EndDate',
        duration: 'Duration',
    }
});

```

```

        progress: 'Progress',
        dependency: 'Predecessor',
        resourceInfo: 'resources',
        work: 'work',
        expandState: 'isExpand',
        child: 'subtasks'
    },
    resourceFields: {
        id: 'resourceId',
        name: 'resourceName',
        unit: 'resourceUnit',
        group: 'resourceGroup'
    },
    editSettings: {
        allowAdding: true,
        allowEditing: true,
        allowDeleting: true,
        allowTaskbarEditing: true,
        showDeleteConfirmDialog: true
    },
    columns: [
        { field: 'TaskID' },
        { field: 'TaskName', headerText: 'Name', width: 250 },
        { field: 'work', headerText: 'Work' },
        { field: 'Progress' },
        { field: 'resourceGroup', headerText: 'Group' },
        { field: 'StartDate' },
        { field: 'Duration' }
    ],
    toolbar: ['ExpandAll', 'CollapseAll'],
    labelSettings: {
        rightLabel: 'resources',
        taskLabel: 'TaskName'
    },
    projectStartDate: new Date('03/28/2019'),
    projectEndDate: new Date('05/18/2019')
});
ganttt.appendTo('#Gantt');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>

<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Holidays in ##Platform_Name## Gantt control

Non-working days in a project can be displayed in the Gantt control using the [holidays](#) property. Each holiday can be defined with the following properties:

- [from](#): Defines start date of the holiday(s).
- [to](#): Defines end date of the holiday(s).
- [label](#): Defines the description or label for the holiday.
- [cssClass](#): Formats the holidays label in the Gantt chart.

To highlight the holidays, inject the [DayMarkers](#) module into the Gantt control.

The following code example shows how to display the non-working days in the Gantt control.

INDEX.JS

```

ej.gantt.Gantt.Inject(ej.gantt.DayMarkers);
var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    holidays: [{
        from: "04/04/2019",
        to: "04/05/2019",
        label: " Public holidays",
        cssClass: "e-custom-holiday"
    },
    {
        from: "04/12/2019",
        to: "04/12/2019",

```



```

        label: " Public holiday",
        cssClass:"e-custom-holiday"

    }]
});
ganttChart.appendTo('#Gantt');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <style>
    .e-gantt .e-gantt-chart .e-custom-holiday {
      background-color:#e82869;
    }
  </style>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Tooltip in ##Platform_Name## Gantt control

The Gantt control has a support to display a tooltip for various UI elements like taskbar, timeline cells, and grid cells

Enable tooltip

In the Gantt control, you can enable or disable the mouse hover tooltip for the following UI elements using the [tooltipSettings.showTooltip](#) property:

- Taskbar
- Connector line
- Baseline
- Event marker

INDEX.JS

```

ej.gantt.Gantt.Inject(ej.gantt.DayMarkers);
var ganttData = [
    {
        TaskID: 1,
        TaskName: 'Project Initiation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        subtasks: [
            { TaskID: 2, TaskName: 'Identify Site
location',BaselineStartDate: new Date('04/02/2019'),BaselineEndDate: new
Date('04/02/2019'), StartDate: new Date('04/02/2019'), Duration: 0,
Progress: 50 },
            { TaskID: 3, TaskName: 'Perform Soil test', StartDate: new
Date('04/02/2019'),BaselineStartDate: new
Date('04/04/2019'),BaselineEndDate: new Date('04/09/2019'), Duration: 4,
Progress: 50 },
            { TaskID: 4, TaskName: 'Soil test approval', StartDate: new
Date('04/02/2019'),BaselineStartDate: new
Date('04/08/2019'),BaselineEndDate: new Date('04/12/2019'), Duration:
4,Predecessor:"2FS", Progress: 50 },
        ]
    },
    {
        TaskID: 5,
        TaskName: 'Project Estimation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        subtasks: [
            { TaskID: 6, TaskName: 'Develop floor plan for
estimation',BaselineStartDate: new Date('04/04/2019'),BaselineEndDate: new
Date('04/08/2019'), StartDate: new Date('04/04/2019'), Duration: 3,
Progress: 50 },
            { TaskID: 7, TaskName: 'List materials', StartDate: new
Date('04/04/2019'),BaselineStartDate: new
Date('04/02/2019'),BaselineEndDate: new Date('04/04/2019'), Duration: 3,
Progress: 50 },
            { TaskID: 8, TaskName: 'Estimation approval', StartDate: new
Date('04/02/2019'),BaselineStartDate: new
Date('04/02/2019'),BaselineEndDate: new Date('04/08/2019'), Duration:
0,Predecessor:"6SS", Progress: 50 }
        ]
    },
];

var ganttChart = new ej.gantt.Gantt({
    dataSource: ganttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',

```

```

        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        baselineStartDate: "BaselineStartDate",
        baselineEndDate: "BaselineEndDate",
        progress: 'Progress',
        dependency: 'Predecessor',
        child: 'subtasks'
    },
    eventMarkers: [
        {
            day: '04/10/2019',
            label: 'Project approval and kick-off'
        }
    ],
    renderBaseline: true,
    baselineColor: 'red',
    tooltipSettings: {
        showTooltip: true
    }
});
gantChart.appendTo('#Gantt');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>
```

The default value of the [tooltipSettings.showTooltip](#) property is true.

Timeline cells tooltip

In the Gantt control, you can enable or disable the mouse hover tooltip of timeline cells using the [timelineSettings.showTooltip](#) property. The default value of this property is true. The following code example shows how to enable the timeline cells tooltip in Gantt.

INDEX.JS

```
var ganttChart = new ej.gantt.Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  timelineSettings: {
    showTooltip: true
  }
});
ganttChart.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
```

```

    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Cell tooltip

You can enable or disable the Grid cell tooltip using the [columns.clipMode](#) property.

INDEX.JS

```

var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    columns: [
        { field: 'TaskID', headerText: 'Task ID', textAlign: 'Left',
width: '100' },
        { field: 'TaskName', headerText: 'Task Name', width: '150',
clipMode: 'EllipsisWithTooltip' },
        { field: 'StartDate', headerText: 'Start Date', width: '150' },
        { field: 'Duration', headerText: 'Duration', width:
'150', clipMode: 'Clip' },
        { field: 'Progress', headerText: 'Progress', width: '150' }
    ]
});
ganttChart.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```

```

    <div id="container">
      <div id="Gantt"></div>
    </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Clip mode

The clip mode provides options to display its overflow cell content and it can be defined by the [columns.clipMode](#) property.

The following are three types of `clipMode`:

- **Clip**: Truncates the cell content when it overflows its area.
- **Ellipsis**: Displays ellipsis when content of the cell overflows its area.
- **EllipsisWithTooltip**: Displays ellipsis when content of the cell overflows its area; it displays the tooltip content when hover over ellipsis.

NOTE

By default, all the column's [clipMode](#) property is defined as **EllipsisWithTooltip**.

Tooltip template

Taskbar tooltip

The default tooltip in the Gantt control can be customized using the [tooltipSettings.taskbar](#) property. You can map the template script element's ID value or template string directly to this property.

INDEX.JS

```

var ganttChart = new ej.gantt.Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  tooltipSettings: {
    showTooltip: true,
    taskbar: '#taskbarTooltip'
  }
});
ganttChart.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <script type="text/x-jsrender" id="taskbarTooltip">
    <div>TaskID: ${TaskID}</div>
  </script>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Connector line tooltip

The default connector line tooltip in the Gantt control can be customized using the [tooltipSettings.connectorLine](#) property. You can map the value to this property as template script element ID or template string format. The following code example shows how to use the [tooltipSettings.connectorLine](#) property.

INDEX.JS

```

var ganttChart = new ej.gantt.Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    dependency: 'Predecessor',
    child: 'subtasks'
  },

```

```

        tooltipSettings: {
            showTooltip: true,
            connectorLine: '#dependencyLineTooltip'
        }
    });
    ganttChart.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <script type="text/x-jsrender" id="dependencyLineTooltip">
        <div>Offset : ${offsetString}</div>
    </script>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Taskbar editing tooltip

The taskbar editing tooltip can be customized using the [tooltipSettings.editing](#) property. The following code example shows how to customize the taskbar editing tooltip in Gantt.

INDEX.JS

```

ej.gantt.Gantt.Inject(ej.gantt.Edit);
var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {

```



```

        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    editSettings: {
        allowEditing: true,
        allowTaskbarEditing: true
    },
    tooltipSettings: {
        showTooltip: true,
        editing: '#editingTooltip'
    }
});
gantChart.appendTo('#Gantt');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <script type="text/x-jsrender" id="editingTooltip">
        <div>Duration : ${duration}</div>
    </script>

    <div id="container">
        <div id="Gantt"></div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>
```

Baseline tooltip

A baseline tooltip can be customized using the [tooltipSettings.baseline](#) property. The following code example shows how to customize the baseline tooltip in Gantt.

INDEX.JS

```
var GanttData = [
    {
        TaskID: 1,
        TaskName: 'Project Initiation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        subtasks: [
            { TaskID: 2, TaskName: 'Identify Site location', BaselineStartDate: new Date('04/02/2019'), BaselineEndDate: new Date('04/02/2019'), StartDate: new Date('04/02/2019'), Duration: 0, Progress: 50 },
            { TaskID: 3, TaskName: 'Perform Soil test', StartDate: new Date('04/02/2019'), BaselineStartDate: new Date('04/04/2019'), BaselineEndDate: new Date('04/09/2019'), Duration: 4, Progress: 50 },
            { TaskID: 4, TaskName: 'Soil test approval', StartDate: new Date('04/02/2019'), BaselineStartDate: new Date('04/08/2019'), BaselineEndDate: new Date('04/12/2019'), Duration: 4, Progress: 50 },
        ]
    },
    {
        TaskID: 5,
        TaskName: 'Project Estimation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        subtasks: [
            { TaskID: 6, TaskName: 'Develop floor plan for estimation', BaselineStartDate: new Date('04/04/2019'), BaselineEndDate: new Date('04/08/2019'), StartDate: new Date('04/04/2019'), Duration: 3, Progress: 50 },
            { TaskID: 7, TaskName: 'List materials', StartDate: new Date('04/04/2019'), BaselineStartDate: new Date('04/02/2019'), BaselineEndDate: new Date('04/04/2019'), Duration: 3, Progress: 50 },
            { TaskID: 8, TaskName: 'Estimation approval', StartDate: new Date('04/02/2019'), BaselineStartDate: new Date('04/02/2019'), BaselineEndDate: new Date('04/08/2019'), Duration: 0, Progress: 50 }
        ]
    }
];

var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
```

```

        baselineStartDate:"BaselineStartDate",
        baselineEndDate:"BaselineEndDate",
        progress: 'Progress',
        child: 'subtasks'
    },
    tooltipSettings: {
        showTooltip: true,
        baseline: '#baselineTooltip'
    },
    renderBaseline:true,
    baselineColor:'red'
});
ganttChart.appendTo('#Gantt');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <script type="text/x-jsrender" id="baselineTooltip">
    <div>Baseline StartDate :
    ${this.getFormattedDate(BaselineStartDate)}</div>
  </script>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Appearance customization in ##Platform_Name## Gantt control

Taskbar customization

Taskbar Height

Height of child taskbars and parent taskbars can be customized by using [taskbarHeight](#) property. The following code example shows how to use the property.

INDEX.JS

```
var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    taskbarHeight: 50,
    rowHeight: 60
});
ganttChart.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
```

```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

NOTE

The [taskbarHeight](#) value should be lower than the [rowHeight](#) property value and these properties accept only pixel values.

Conditional formatting

The default taskbar UI can be replaced with custom templates using the [queryTaskbarInfo](#) event. The following code example shows customizing the taskbar UI based on task progress values in the Gantt control.

INDEX.JS

```
var GanttData = [
    {
        TaskID: 1,
        TaskName: 'Project Initiation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        subtasks: [
            { TaskID: 2, TaskName: 'Identify Site location', StartDate:
new Date('04/02/2019'), Duration: 4, Progress: 50 },
            { TaskID: 3, TaskName: 'Perform Soil test', StartDate: new
Date('04/02/2019'), Duration: 4, Progress: 70 },
            { TaskID: 4, TaskName: 'Soil test approval', StartDate: new
Date('04/02/2019'), Duration: 4, Progress: 80},
        ]
    },
    {
        TaskID: 5,
        TaskName: 'Project Estimation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        subtasks: [
            { TaskID: 6, TaskName: 'Develop floor plan for estimation',
StartDate: new Date('04/04/2019'), Duration: 3, Progress: 50 },
            { TaskID: 7, TaskName: 'List materials', StartDate: new
Date('04/04/2019'), Duration: 3, Progress: 50 },
            { TaskID: 8, TaskName: 'Estimation approval', StartDate: new
Date('04/04/2019'), Duration: 3, Progress: 70 }
        ]
    },
];
var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
});
```

```

        queryTaskbarInfo: function(args) {
            if (args.data.Progress == 50) {
                args.progressBarBgColor = "red";
            } else if (args.data.Progress == 70) {
                args.progressBarBgColor = "yellow";
            } else if (args.data.Progress == 80) {
                args.progressBarBgColor = "lightgreen";
            }
        }
    });
    ganttChart.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Taskbar template

You can design your own taskbars to view the tasks in Gantt by using [taskbarTemplate](#) property. And it is possible to map the template script element's ID value to this property. It is also possible to customize the parent taskbars and milestones with custom templates by using [parentTaskbarTemplate](#) and [milestoneTemplate](#) properties.

INDEX.JS

```

var ganttChart = new ej.gantt.Gantt({

```

```

        dataSource: GanttData,
        height: '450px',
        taskFields: {
            id: 'TaskID',
            name: 'TaskName',
            startDate: 'StartDate',
            duration: 'Duration',
            progress: 'Progress',
            child: 'subtasks'
        },
        rowHeight: 60,
        milestoneTemplate: '#MilestoneTemplate',
        parentTaskbarTemplate: '#ParentTaskbarTemplate',
        taskbarTemplate: '#TaskbarTemplate'
    });
    ganttChart.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <script type="text/x-jsrender" id="TaskbarTemplate">
        <div class="e-gantt-child-taskbar-inner-div e-gantt-child-taskbar"
style="height:100%">
            <div class="e-gantt-child-progressbar-inner-div e-gantt-child-
progressbar" style="width:${ganttProperties.progressWidth}px;height:100%">
                <span class="e-task-label" style="position: absolute; z-index: 1;
font-size: 12px; color: white; top: 5px; left: 10px; font-family: "Segoe
UI"; overflow: hidden; text-overflow: ellipsis; width: 40%; cursor:
move;">${taskData.TaskName}</span>
            </div>
        </div>
    </script>
    <script type="text/x-jsrender" id="ParentTaskbarTemplate">
        <div class="e-gantt-parent-taskbar-inner-div e-gantt-parent-taskbar"
style="height:100%">
            <div class="e-gantt-parent-progressbar-inner-div e-gantt-parent-
progressbar" style="width:${ganttProperties.progressWidth}px;height:100%">
                <span class="e-task-label" style="position: absolute; z-index: 1;
font-size: 12px; color: white; top: 5px; left: 10px; font-family: "Segoe

```

```

UI"; overflow: hidden; text-overflow: ellipsis; width: 40%; cursor:
move;">${taskData.TaskName}</span>
</div>
</div>
</script>
<script type="text/x-jsrender" id="MilestoneTemplate">
  <div class="e-gantt-milestone" style="position:absolute;">
    <div class="e-milestone-top" style="border-right-width:15px;border-
left-width:15px;border-bottom-width:15px;"></div>
    <div class="e-milestone-bottom" style="top:15px;border-right-
width:15px; border-left-width:15px; border-top-width:15px;">
      </div>
    </div>
  </script>

  <div id="container">
    <div id="Gantt"></div>
  </div>
</script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Change gripper icon in taskbar

You can change the gripper icon in the taskbar by applying styles to their respective class elements.

INDEX.JS

```

ej.gantt.Gantt.Inject(ej.gantt.Edit, ej.gantt.Sort, ej.gantt.Filter);
var projectResources= [
  { resourceId: 1, resourceName: 'Martin Tamer' },
  { resourceId: 2, resourceName: 'Rose Fuller' },
  { resourceId: 3, resourceName: 'Margaret Buchanan' },
  { resourceId: 4, resourceName: 'Fuller King' },
  { resourceId: 5, resourceName: 'Davolio Fuller' },
  { resourceId: 6, resourceName: 'Van Jack' },
  { resourceId: 7, resourceName: 'Fuller Buchanan' },
  { resourceId: 8, resourceName: 'Jack Davolio' },
  { resourceId: 9, resourceName: 'Tamer Vinet' },
  { resourceId: 10, resourceName: 'Vinet Fuller' },
  { resourceId: 11, resourceName: 'Bergs Anton' },
  { resourceId: 12, resourceName: 'Construction Supervisor' }
];
var ganttData = [
  {
    TaskID: 1,
    TaskName: 'Project Initiation',
    StartDate: new Date('04/02/2019'),
    EndDate: new Date('04/21/2019'),
    subtasks: [
      { TaskID: 2, TaskName: 'Identify Site location', StartDate:
new Date('04/02/2019'), Duration: 0, Progress: 50 },

```



```

        { TaskID: 3, TaskName: 'Perform Soil test', StartDate: new
Date('04/02/2019'), Duration: 4, Progress: 50, resources: [2, 3, 5] },
        { TaskID: 4, TaskName: 'Soil test approval', StartDate: new
Date('04/02/2019'), Duration: 4,Predecessor:"2FS", Progress: 50 },
    ]
},
{
    TaskID: 5,
    TaskName: 'Project Estimation',
    StartDate: new Date('04/02/2019'),
    EndDate: new Date('04/21/2019'),
    subtasks: [
        { TaskID: 6, TaskName: 'Develop floor plan for estimation',
StartDate: new Date('04/04/2019'), Duration: 3, Progress: 50, resources: [4]
},
        { TaskID: 7, TaskName: 'List materials', StartDate: new
Date('04/04/2019'), Duration: 3, Progress: 50, resources: [4, 8], },
        { TaskID: 8, TaskName: 'Estimation approval', StartDate: new
Date('04/04/2019'), Duration: 0,Predecessor:"6SS", Progress: 50, resources:
[12, 5] }
    ]
},
];
var ganttChart = new ej.gantt.Gantt({
    dataSource: ganttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        resourceInfo: 'resources',
        duration: 'Duration',
        progress: 'Progress',
        dependency: 'Predecessor',
        child: 'subtasks'
    },
    columns: [
        { field: 'TaskID', headerText: 'Task ID', width: '100' },
        { field: 'TaskName', headerText: 'Task Name', width: '250' },
        { field: 'StartDate', headerText: 'Start Date', width: '150' },
        { field: 'resources', headerText: 'Resources', width:
'200' },
        { field: 'Duration', headerText: 'Duration', width: '150' },
        { field: 'Progress', headerText: 'Progress', width: '150' },
    ],
    resourceFields: {
        id: 'resourceId',
        name: 'resourceName',
    },
    resources: projectResources,
    allowSorting: true,
    editSettings: {
        allowEditing: true,
        allowTaskbarEditing: true
    },
    allowFiltering: true
});

```

```
ganttChart.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Task labels

The Gantt control maps any data source fields to task labels using the [labelSettings.leftLabel](#), [labelSettings.rightLabel](#), and [labelSettings.taskLabel](#) properties. You can customize the task labels with templates.

INDEX.JS

```
var ganttChart = new ej.gantt.Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  labelSettings: {
```

```

        leftLabel: 'TaskID',
        rightLabel: 'Task Name: ${taskData.TaskName}',
        taskLabel: '${Progress}%'
    }
});
gantChart.appendTo('#Gantt');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Connector lines

The width and background color of connector lines in Gantt can be customized using the [connectorLineWidth](#) and [connectorLineBackground](#) properties. The following code example shows how to use these properties.

INDEX.JS

```

var GanttData = [
    {
        TaskID: 1,
        TaskName: 'Project Initiation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        subtasks: [
```

```

        { TaskID: 2, TaskName: 'Identify Site location', StartDate:
new Date('04/02/2019'), Duration: 4, Progress: 50 },
        { TaskID: 3, TaskName: 'Perform Soil test', StartDate: new
Date('04/02/2019'), Duration: 4, Progress: 50 },
        { TaskID: 4, TaskName: 'Soil test approval', StartDate: new
Date('04/02/2019'),Predecessor:"3SS", Duration: 4, Progress: 50 },
    ]
},
{
    TaskID: 5,
    TaskName: 'Project Estimation',
    StartDate: new Date('04/02/2019'),
    EndDate: new Date('04/21/2019'),
    subtasks: [
        { TaskID: 6, TaskName: 'Develop floor plan for estimation',
StartDate: new Date('04/04/2019'), Duration: 3, Progress: 50 },
        { TaskID: 7, TaskName: 'List materials', StartDate: new
Date('04/04/2019'), Duration: 3, Progress: 50 },
        { TaskID: 8, TaskName: 'Estimation approval', StartDate: new
Date('04/04/2019'),Predecessor:"7FS", Duration: 3, Progress: 50 }
    ]
},
];

var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        dependency: 'Predecessor',
        progress: 'Progress',
        child: 'subtasks'
    },
    connectorLineBackground: "red",
    connectorLineWidth: 3
});
ganttChart.appendTo('#Gantt');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
```

```

<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customize rows and cells

While rendering the TreeGrid part in Gantt, the [rowDataBound](#) and [queryCellInfo](#) events trigger for every row and cell. Using these events, you can customize the rows and cells. The following code example shows how to customize the cell and row elements using these events.

INDEX.JS

```

var GanttData = [
    {
        TaskID: 1,
        TaskName: 'Project Initiation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        subtasks: [
            { TaskID: 2, TaskName: 'Identify Site location', StartDate:
new Date('04/02/2019'), Duration: 0, Progress: 80 },
            { TaskID: 3, TaskName: 'Perform Soil test', StartDate: new
Date('04/02/2019'), Duration: 4, Progress: 50 },
            { TaskID: 4, TaskName: 'Soil test approval', StartDate: new
Date('04/02/2019'), Duration: 4, Progress: 50 },
        ]
    },
    {
        TaskID: 5,
        TaskName: 'Project Estimation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        subtasks: [
            { TaskID: 6, TaskName: 'Develop floor plan for estimation',
StartDate: new Date('04/04/2019'), Duration: 3, Progress: 50 },
            { TaskID: 7, TaskName: 'List materials', StartDate: new
Date('04/04/2019'), Duration: 3, Progress: 60 },
            { TaskID: 8, TaskName: 'Estimation approval', StartDate: new
Date('04/04/2019'), Duration: 0, Progress: 50 }
        ]
    },
];

```

```

var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        dependency: 'Predecessor',
        progress: 'Progress',
        child: 'subtasks'
    },
    columns: [
        { field: 'TaskID', headerText: 'Task ID', textAlign: 'Left',
width: '100' },
        { field: 'TaskName', headerText: 'Task Name', width: '150' },
            { field: 'Progress', headerText: 'Progress', width:
'150' },
            { field: 'StartDate', headerText: 'Start Date',
width: '150' },
            { field: 'Duration', headerText: 'Duration', width: '150' },
    ],
    splitterSettings: {
        columnIndex: 3
    },
    queryCellInfo: function (args) {
        if (args.column.field == "Progress") {
            if (args.data.Progress < 60)
                args.cell.style.backgroundColor="lightgreen"
            else
                args.cell.style.backgroundColor="yellow"
        }
    },
    rowDataBound: function (args) {
        if (args.data.TaskID==4)
            args.row.style.backgroundColor="red"
    }
});
ganttChart.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>

```

```

<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Grid lines

In the Gantt control, you can show or hide the grid lines in the TreeGrid side and chart side by using the [gridLines](#) property.

The following options are available in the Gantt control for rendering the grid lines:

- Horizontal: The horizontal grid lines alone will be visible.
- Vertical: The vertical grid lines alone will be visible.
- Both: Both the horizontal and vertical grid lines will be visible on the TreeGrid and chart sides.
- None: Gridlines will not be visible on TreeGrid and chart sides.

By default, the [gridLines](#) property is set to **Horizontal** type.

The following code example shows how to change the gridlines rendering mode in the Gantt control.

INDEX.JS

```

var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    gridLines: 'Both'
});
ganttChart.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>

```

```

<title>EJ2 Gantt</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Gantt Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Splitter

In the Gantt control, the Splitter separates the TreeGrid section from the Chart section. You can change the position of the Splitter when loading the Gantt control using the [splitterSettings](#) property. By splitting the TreeGrid from the chart, the width of the TreeGrid and chart sections will vary in the control. The [splitterSettings.position](#) property denotes the percentage of the TreeGrid section's width to be rendered and this property supports both pixels and percentage values. You can define the splitter position as column index value using the [splitterSettings.columnIndex](#) property. You can also define the splitter position with built-in splitter view modes by using the [splitterSettings.view](#) property. The following list is the possible values for this property:

- **Default:** Shows Grid side and Gantt side.
- **Grid:** Shows Grid side alone in Gantt.
- **Chart:** Shows chart side alone in Gantt.

INDEX.JS

```

var ganttData = [
    {
        TaskID: 1,
        TaskName: 'Project Initiation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
    }
]

```



```

        isParent:true,
        subtasks: [
            { TaskID: 2, TaskName: 'Identify Site location', StartDate:
new Date('04/02/2019'), Duration: 3, Progress: 50,isParent:false },
            { TaskID: 3, TaskName: 'Perform Soil test', StartDate: new
Date('04/02/2019'), Duration: 3, Progress: 70, resources: [2, 3,
5],isParent:false },
            { TaskID: 4, TaskName: 'Soil test approval', StartDate: new
Date('04/02/2019'), Duration: 3, Predecessor:"2FS", Progress:
80,isParent:false },
        ]
    },
    {
        TaskID: 5,
        TaskName: 'Project Estimation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        isParent:true,
        subtasks: [
            { TaskID: 6, TaskName: 'Develop floor plan for estimation',
StartDate: new Date('04/04/2019'), Duration: 4, Progress: 50, resources:
[4],isParent:false },
            { TaskID: 7, TaskName: 'List materials', StartDate: new
Date('04/04/2019'), Duration: 4, Progress: 50, DurationUnit:'day',
resources: [4, 8],isParent:false },
            { TaskID: 8, TaskName: 'Estimation approval', StartDate: new
Date('04/04/2019'), Duration: 4,Predecessor:"6SS", DurationUnit:'minute',
Progress: 70, resources: [12, 5],isParent:false }
        ]
    },
];
var ganttChart = new ej.gantt.Gantt({
    dataSource: ganttData,
    height:'450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    splitterSettings:{
        position: "50%"
    }
});
ganttChart.appendTo('#Gantt');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
```

```

<link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Change splitter position dynamically

In Gantt, we can change the splitter position dynamically by using [setSplitterPosition](#) method. We can change the splitter position by passing value and type parameter to [setSplitterPosition](#) method. Type parameter will accept one of the following values 'position', 'columnIndex', 'viewType'. The following code example shows how to use this method.

INDEX.JS

```

var ganttData = [
  {
    TaskID: 1,
    TaskName: 'Project Initiation',
    StartDate: new Date('04/02/2019'),
    EndDate: new Date('04/21/2019'),
    isParent: true,
    subtasks: [
      { TaskID: 2, TaskName: 'Identify Site location', StartDate:
new Date('04/02/2019'), Duration: 3, Progress: 50, isParent: false },
      { TaskID: 3, TaskName: 'Perform Soil test', StartDate: new
Date('04/02/2019'), Duration: 3, Progress: 70, resources: [2, 3,
5], isParent: false },
      { TaskID: 4, TaskName: 'Soil test approval', StartDate: new
Date('04/02/2019'), Duration: 3, Predecessor: "2FS", Progress:
80, isParent: false },
    ]
  },
  {
    TaskID: 5,
    TaskName: 'Project Estimation',
    StartDate: new Date('04/02/2019'),
    EndDate: new Date('04/21/2019'),

```

```

        isParent:true,
        subtasks: [
            { TaskID: 6, TaskName: 'Develop floor plan for estimation',
StartDate: new Date('04/04/2019'), Duration: 4, Progress: 50, resources:
[4],isParent:false },
            { TaskID: 7, TaskName: 'List materials', StartDate: new
Date('04/04/2019'), Duration: 4, Progress: 50, DurationUnit:'day',
resources: [4, 8],isParent:false },
            { TaskID: 8, TaskName: 'Estimation approval', StartDate: new
Date('04/04/2019'), Duration: 4,Predecessor:"6SS", DurationUnit:'minute',
Progress: 70, resources: [12, 5],isParent:false }
        ]
    }
};
var ganttChart = new ej.gantt.Gantt({
    dataSource: ganttData,
    height:'450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    }
});
ganttChart.appendTo('#Gantt');
var splitBtn= new ej.buttons.Button();
splitBtn.appendTo('#changeByPosition');
var splittBtn= new ej.buttons.Button();
splittBtn.appendTo('#changeByIndex');
document.getElementById('changeByPosition').addEventListener('click',
function() {
    ganttChart.setSplitterPosition('50%', 'position');
});
document.getElementById('changeByIndex').addEventListener('click',
function() {
    ganttChart.setSplitterPosition(0, 'columnIndex');
});
var dropDownMode = new ej.dropdowns.DropDownList({
    dataSource: [
        { id: 'Default', mode: 'Default' },
        { id: 'Grid', mode: 'Grid' },
        { id: 'Chart', mode: 'Chart' },
    ],
    fields: { text: 'mode', value: 'id' },
    value: 'Default',
    change: function (e) {
        var viewType = e.value;
        ganttChart.setSplitterPosition(viewType, 'view');
    }
});
dropDownMode.appendTo('#view');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">

    <div>
      <div style="padding-top: 7px; display: inline-block">Change Splitter
View</div>
      <div style="display: inline-block">
        <input type="text" id="view">
      </div>
    </div>

    <button id="changeByPosition">Change Splitter By
Position</button> <button id="changeByIndex">Change Splitter By
ColumnIndex</button>
    <div id="Gantt"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Duration unit in ##Platform_Name## Gantt control

Duration units

In Gantt, the task's duration value can be measured by the following duration units,

- Day
- Hour
- Minute

In Gantt, we can define duration unit for whole project by using [durationUnit](#) property, when we defines the value for this property, this unit will be applied for all task which don't has duration unit value. And

each task in the project can be defined with different duration units and the duration unit of a task can be defined by the following ways,

- Using [taskFields.durationUnit](#) property, to map the duration unit data source field.
- Defining the duration unit value along with the duration field in the data source.

Mapping the duration unit field

The below code snippet explains the mapping of duration unit data source field to the Gantt control using the [taskFields.durationUnit](#) property.

INDEX.JS

```
var ganttData = [
    {
        TaskID: 1,
        TaskName: 'Project Initiation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        subtasks: [
            { TaskID: 2, TaskName: 'Identify Site location', StartDate:
new Date('04/02/2019'), Duration: 2, DurationUnit:'day', Progress: 50 },
            { TaskID: 3, TaskName: 'Perform Soil test', StartDate: new
Date('04/02/2019'), Duration: 12,DurationUnit:'hour', Progress: 70 },
            { TaskID: 4, TaskName: 'Soil test approval', StartDate: new
Date('04/02/2019'), Duration: 240,DurationUnit:'minute', Progress: 80 },
        ]
    },
    {
        TaskID: 5,
        TaskName: 'Project Estimation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        subtasks: [
            { TaskID: 6, TaskName: 'Develop floor plan for estimation',
StartDate: new Date('04/04/2019'), Duration: 3, DurationUnit:'hour',
Progress: 50 },
            { TaskID: 7, TaskName: 'List materials', StartDate: new
Date('04/04/2019'), Duration: 3, Progress: 50, DurationUnit:'day' },
            { TaskID: 8, TaskName: 'Estimation approval', StartDate: new
Date('04/04/2019'), Duration: 480, DurationUnit:'minute', Progress: 70 }
        ]
    },
];
var ganttChart = new ej.gantt.Gantt({
    dataSource: ganttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        durationUnit: 'DurationUnit',
        child: 'subtasks'
    },
    splitterSettings: {
```

```

        columnIndex: 4
    }
});
gantChart.appendTo('#Gantt');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

NOTE

The default value of the [durationUnit](#) property is `day`.

Defining duration unit along with duration field

Duration units for the tasks can also be defined along with the duration values, the below code snippet explains the duration unit for a task along with duration value,

INDEX.JS

```

var ganttData = [
  {
    TaskID: 1,
    TaskName: 'Project Initiation',
    StartDate: new Date('04/02/2019'),
    EndDate: new Date('04/21/2019'),
    isParent: true,
    subtasks: [
```

```

        { TaskID: 2, TaskName: 'Identify Site location', StartDate:
new Date('04/02/2019'), Duration: '3days', Progress: 50 },
        { TaskID: 3, TaskName: 'Perform Soil test', StartDate: new
Date('04/02/2019'), Duration: '12hours', Progress: 70 },
        { TaskID: 4, TaskName: 'Soil test approval', StartDate: new
Date('04/02/2019'), Duration: '1800minutes', Progress: 80 },
    ]
},
{
    TaskID: 5,
    TaskName: 'Project Estimation',
    StartDate: new Date('04/02/2019'),
    EndDate: new Date('04/21/2019'),
    isParent:true,
    subtasks: [
        { TaskID: 6, TaskName: 'Develop floor plan for estimation',
StartDate: new Date('04/04/2019'), Duration: '12hours',Progress: 50 },
        { TaskID: 7, TaskName: 'List materials', StartDate: new
Date('04/04/2019'), Duration: '3days', Progress: 50 },
        { TaskID: 8, TaskName: 'Estimation approval', StartDate: new
Date('04/04/2019'), Duration: '480minutes', Progress: 70 }
    ]
},
];
var ganttChart = new ej.gantt.Gantt({
    dataSource: ganttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    splitterSettings:{
        columnIndex:4
    }
});
ganttChart.appendTo('#Gantt');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

NOTE:

The edit type of the duration column in Gantt is string, to support editing the duration field along with duration units.

Task scheduling in ##Platform_Name## Gantt control

The Gantt provides support for automatic and manual task scheduling modes. It is used to indicate whether the start date and end date of all the tasks will be automatically validated or not. [taskMode](#) is the property used to change the schedule mode of a task.

The Gantt control supports three types of mode. They are:

- **Auto**: All the tasks are automatically validate.
- **Manual**: All the tasks are manually validate by the user.
- **Custom**: Both Auto and Manual tasks are render by mapped from data source.

Note: The default value of [taskMode](#) is **Auto**.

Automatically Scheduled Tasks

When the [taskMode](#) property is set as **Auto**, the start date and end date of all the tasks in the project will be automatically validated. That is, dates are validated based on various factors such as working time, holidays, weekends and predecessors.

INDEX.JS

```

ej.gantt.Gantt.Inject(ej.gantt.Edit,ej.gantt.Toolbar,ej.gantt.Selection);
var gantt = new ej.gantt.Gantt({
    dataSource: GanttData,
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        endDate: 'EndDate',

```



```

        dependency: 'Predecessor',
        child: 'Children',
    },
    height: '450px',
    taskMode : 'Auto',
    toolbar: ['Add', 'Edit', 'Update', 'Delete', 'Cancel', 'ExpandAll',
'CollapseAll', 'Search'],
    treeColumnIndex: 1,
    editSettings: {
        allowEditing: true,
        allowDeleting: true,
        allowTaskbarEditing: true,
        showDeleteConfirmDialog: true
    },
    });
ganttt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>

    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

Manually Scheduled Tasks

When the [taskMode](#) property is set as **Manual**, the start date and end date of all the tasks in the project will be same as given in the data source. That is, dates are not validated based on various factors such as

dependencies between tasks, holidays, weekends, working time. We can restrict this mode in predecessor validation alone. That is, we can automatically validate the dates based on predecessor values by enabling the [validateManualTasksOnLinking](#) property.

INDEX.JS

```
ej.gantt.Gantt.Inject(ej.gantt.Edit,ej.gantt.Toolbar,ej.gantt.Selection);
var gantt = new ej.gantt.Gantt({
  dataSource: GanttData,
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    endDate: 'EndDate',
    dependency: 'Predecessor',
    child: 'Children',
  },
  height: '450px',
  taskMode : 'Manual',
  toolbar: ['Add', 'Edit', 'Update', 'Delete', 'Cancel', 'ExpandAll',
  'CollapseAll', 'Search'],
  treeColumnIndex: 1,
  editSettings: {
    allowEditing: true,
    allowDeleting: true,
    allowTaskbarEditing: true,
    showDeleteConfirmDialog: true
  },
});
gantt.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
```

```

        </div>

<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Custom

When the [taskMode](#) property is set as Custom, the scheduling mode for each tasks will be mapped from the data source field. The [Boolean](#) property [taskFields.manual](#) is used to map the manual scheduling mode field from the data source.

INDEX.JS

```

ej.gantt.Gantt.Inject(ej.gantt.Edit,ej.gantt.Toolbar,ej.gantt.Selection);
var GanttData = [
    {
        'TaskID': 1,
        'TaskName': 'Parent Task 1',
        'StartDate': new Date('02/27/2017'),
        'EndDate': new Date('03/03/2017'),
        'Progress': '40',
        'isManual': true,
        'Children': [
            { 'TaskID': 2, 'TaskName': 'Child Task 1', 'StartDate': new
Date('02/27/2017'),
                'EndDate': new Date('03/03/2017'), 'Progress': '40' },
            { 'TaskID': 3, 'TaskName': 'Child Task 2', 'StartDate': new
Date('02/26/2017'),
                'EndDate': new Date('03/03/2017'), 'Progress': '40',
                'isManual': true },
            { 'TaskID': 4, 'TaskName': 'Child Task 3', 'StartDate': new
Date('02/27/2017'),
                'EndDate': new Date('03/03/2017'), 'Duration': 5, 'Progress':
'40', }
        ]
    },
    {
        'TaskID': 5,
        'TaskName': 'Parent Task 2',
        'StartDate': new Date('03/05/2017'),
        'EndDate': new Date('03/09/2017'),
        'Progress': '40',
        'isManual': true,
        'Children': [
            { 'TaskID': 6, 'TaskName': 'Child Task 1', 'StartDate': new
Date('03/06/2017'),
                'EndDate': new Date('03/09/2017'), 'Progress': '40' },
            { 'TaskID': 7, 'TaskName': 'Child Task 2', 'StartDate': new
Date('03/06/2017'),
                'EndDate': new Date('03/09/2017'), 'Progress': '40', },
        ]
    }
]

```

```

        { 'TaskID': 8, 'TaskName': 'Child Task 3', 'StartDate': new
Date('02/28/2017'),
        'EndDate': new Date('03/05/2017'), 'Progress': '40',
        'isManual': true },
        { 'TaskID': 9, 'TaskName': 'Child Task 4', 'StartDate': new
Date('03/04/2017'),
        'EndDate': new Date('03/09/2017'), 'Progress': '40',
        'isManual': true }
    ]
},
{
    'TaskID': 10,
    'TaskName': 'Parent Task 3',
    'StartDate': new Date('03/13/2017'),
    'EndDate': new Date('03/17/2017'),
    'Progress': '40',
    'Children': [
        { 'TaskID': 11, 'TaskName': 'Child Task 1', 'StartDate': new
Date('03/13/2017'),
        'EndDate': new Date('03/17/2017'), 'Progress': '40' },
        { 'TaskID': 12, 'TaskName': 'Child Task 2', 'StartDate': new
Date('03/13/2017'),
        'EndDate': new Date('03/17/2017'), 'Progress': '40' },
        { 'TaskID': 13, 'TaskName': 'Child Task 3', 'StartDate': new
Date('03/13/2017'),
        'EndDate': new Date('03/17/2017'), 'Progress': '40' },
        { 'TaskID': 14, 'TaskName': 'Child Task 4', 'StartDate': new
Date('03/12/2017'),
        'EndDate': new Date('03/17/2017'), 'Progress': '40',
        'isManual': true },
        { 'TaskID': 15, 'TaskName': 'Child Task 5', 'StartDate': new
Date('03/13/2017'),
        'EndDate': new Date('03/17/2017'), 'Progress': '40' }
    ]
}
];
var gantt = new ej.gantt.Gantt({
    dataSource: GanttData,
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        endDate: 'EndDate',
        dependency: 'Predecessor',
        child: 'Children',
        manual: 'isManual',
    },
    height: '450px',
    taskMode : 'Custom',
    toolbar: ['Add', 'Edit', 'Update', 'Delete', 'Cancel', 'ExpandAll',
'CollapseAll', 'Search'],
    columns: [
        { field: 'TaskID', visible: false},
        {field: 'TaskName'},
        { field: 'isManual'}
    ]
});

```

```

    ],
    treeColumnIndex: 1,
    editSettings: {
        allowEditing: true,
        allowDeleting: true,
        allowTaskbarEditing: true,
        showDeleteConfirmDialog: true
    },
    });
gantt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>

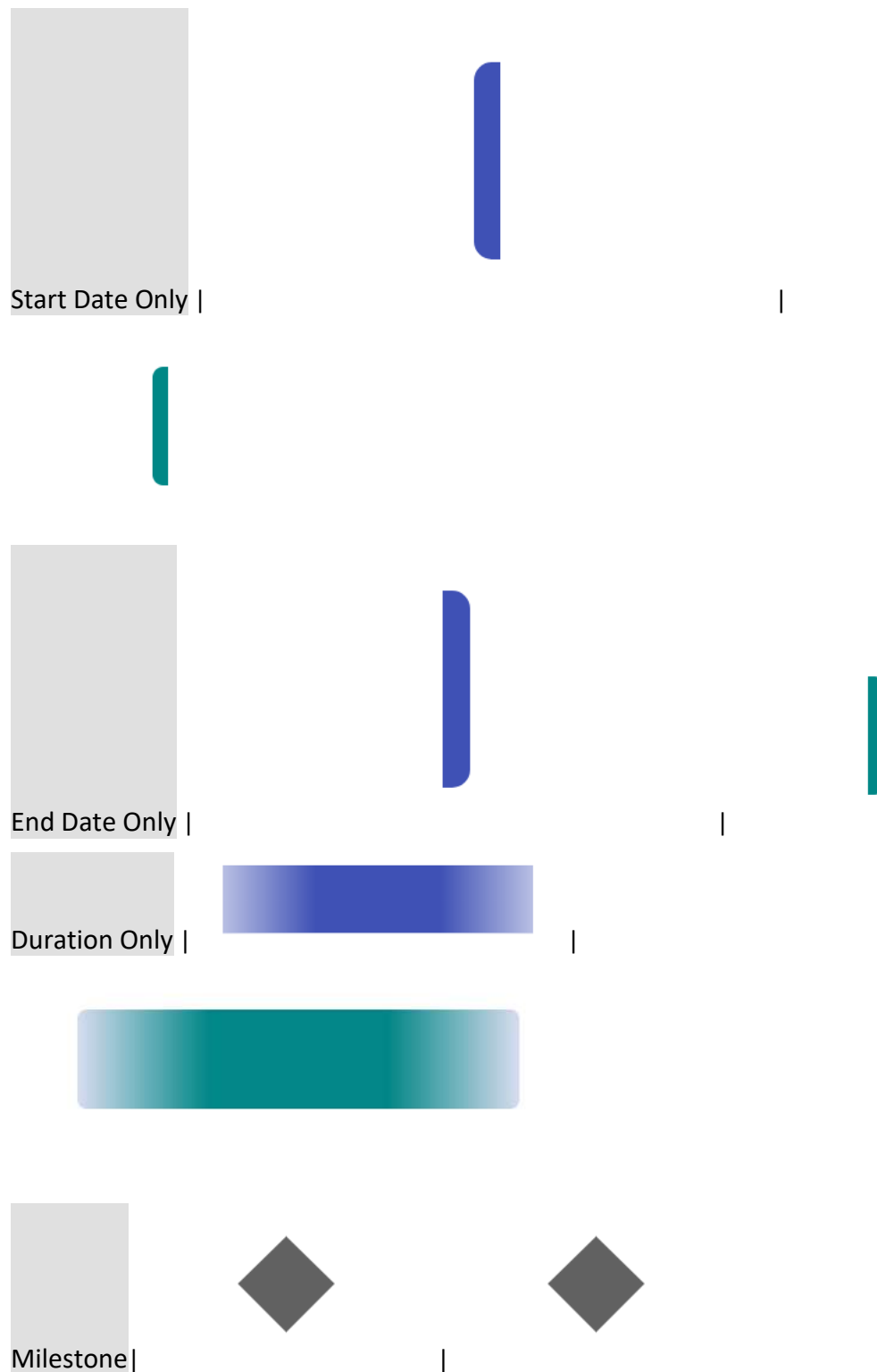
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Unscheduled Tasks

Unscheduled tasks are planned for a project without any definite schedule dates. The Gantt control supports rendering the unscheduled tasks. You can create or update the tasks with anyone of start date, end date, and duration values or none. You can enable or disable the unscheduled tasks by using the [allowUnscheduledTasks](#) property. The following images represent the various types of unscheduled tasks in Gantt.

Taskbar state | Auto | Manual



Note: A milestone is a task that has no start and end dates, but it has a duration value of zero

[Define unscheduled tasks in data source](#)

You can define the various types of unscheduled tasks in the data source as follows

INDEX.JS

```

ej.gantt.Gantt.Inject(ej.gantt.Edit);
var GanttData = [
    {
        TaskID: 1,
        TaskName: 'Project Initiation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        subtasks: [
            { TaskID: 2, TaskName: 'Identify Site location', Duration:
3, Progress: 50},
            { TaskID: 3, TaskName: 'Perform Soil test', StartDate: new
Date('04/02/2019'), Progress: 50 },
            { TaskID: 4, TaskName: 'Soil test approval', EndDate: new
Date('04/08/2019'), Progress: 50 },
        ]
    },
    {
        TaskID: 5,
        TaskName: 'Project Estimation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        subtasks: [
            { TaskID: 6, TaskName: 'Develop floor plan for estimation',
StartDate: new Date('04/04/2019'), EndDate: new Date('04/08/2019'), Progress:
50 },
            { TaskID: 7, TaskName: 'List materials', StartDate: new
Date('04/04/2019'), Progress: 50 },
            { TaskID: 8, TaskName: 'Estimation approval', Duration:
0, Progress: 50}
        ]
    },
];
var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        endDate: 'EndDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    editSettings: {
        allowEditing: true,
        allowTaskbarEditing: true
    },
    allowUnscheduledTasks: true
});
ganttChart.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
<title>EJ2 Gantt</title>

```

```

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Gantt Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

NOTE

If the [allowUnscheduledTasks](#) property is set to false, then the Gantt control automatically calculates the scheduled date values with a default value of duration 1 and the project start date is considered as the start date for the task.

Working Time Range

In the Gantt control, working hours in a day for a project can be defined by using the [dayWorkingTime](#) property. Based on the working hours, automatic date scheduling and duration validations for a task are performed.

The following code snippet explains how to define the working time range for the project in Gantt.

INDEX.JS

```

ej.gantt.Gantt.Inject(ej.gantt.DayMarkers);
var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },

```



```

        highlightWeekends: true,
        dayWorkingTime: [
            { from: 9,
              to: 18 }
        ],
        timelineSettings: {
            timelineViewMode: 'Day'
        },
        splitterSettings: {
            columnIndex: 0
        }
    });
    ganttChart.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

NOTE

- * Individual tasks can lie between any time within the defined working time range of the project.
- * The [dayWorkingTime](#) property is used to define the working time for the whole project.

Weekend/Non-working days

Non-working days/weekend are used to represent the non-productive days in a project. You can define the non-working days in a week using the [workWeek](#) property in Gantt.

INDEX.JS

```
var ganttData = [
    {
        TaskID: 1,
        TaskName: 'Project Initiation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        isParent: true,
        subtasks: [
            { TaskID: 2, TaskName: 'Identify Site location', StartDate:
new Date('04/02/2019'), Duration: 3, Progress: 50, isParent: false },
            { TaskID: 3, TaskName: 'Perform Soil test', StartDate: new
Date('04/02/2019'), Duration: 3, Progress: 70, resources: [2, 3,
5], isParent: false },
            { TaskID: 4, TaskName: 'Soil test approval', StartDate: new
Date('04/02/2019'), Duration: 3, Predecessor: "2FS", Progress:
80, isParent: false },
        ]
    },
    {
        TaskID: 5,
        TaskName: 'Project Estimation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        isParent: true,
        subtasks: [
            { TaskID: 6, TaskName: 'Develop floor plan for estimation',
StartDate: new Date('04/04/2019'), Duration: 4, Progress: 50, resources:
[4], isParent: false },
            { TaskID: 7, TaskName: 'List materials', StartDate: new
Date('04/04/2019'), Duration: 4, Progress: 50, DurationUnit: 'day',
resources: [4, 8], isParent: false },
            { TaskID: 8, TaskName: 'Estimation approval', StartDate: new
Date('04/04/2019'), Duration: 4, Predecessor: "6SS", DurationUnit: 'minute',
Progress: 70, resources: [12, 5], isParent: false },
        ]
    },
];
var ganttChart = new ej.gantt.Gantt({
    dataSource: ganttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    workWeek: ["Sunday", "Monday", "Tuesday", "Wednesday", "Thursday"],
    highlightWeekends: true
});
```

```
ganttChart.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

By default, Saturdays and Sundays are considered as non-working days/weekend in a project.

In the Gantt control, you can make weekend as working day by setting the [includeWeekend](#) property to `true`.

Critical path in ##Platform_Name## Gantt control

The critical path in a project is indicated by a single task or a series of tasks. If a task in critical path is delayed, the entire project will be delayed. A task is considered to be critical if any delay to this task would affect the project end date.

The critical path can be enabled in Gantt by using the built-in toolbar button or [enableCriticalPath](#) property.

The following code example shows how to display the critical path in the Gantt control:

INDEX.JS

```
ej.gantt.Gantt.Inject(ej.gantt.Edit, ej.gantt.Toolbar,
ej.gantt.CriticalPath);
var ganttChart = new ej.gantt.Gantt({
  dataSource: GanttData,
```

```

height: '450px',
taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks',
},
enableCriticalPath: true,
editSettings: {
    allowAdding: true,
    allowEditing: true,
    allowDeleting: true,
    allowTaskbarEditing: true,
    showDeleteConfirmDialog: true
},
toolbar: ['CriticalPath'],
});
gantChart.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <style>
        .e-gantt .e-gantt-chart .e-custom-holiday {
            background-color:#e82869;
        }
    </style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";

```

```

}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customize taskbar in critical path

The taskbar in critical path can be customized by using [queryTaskbarInfo](#) event and [isCritical](#) property of row [data](#) in the event argument.

The following code example shows how to customize the critical path taskbar in the Gantt control:

INDEX.JS

```

ej.gantt.Gantt.Inject(ej.gantt.Edit, ej.gantt.Toolbar,
ej.gantt.CriticalPath);
var ganttChart = new ej.gantt.Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks',
  },
  enableCriticalPath: true,
  editSettings: {
    allowAdding: true,
    allowEditing: true,
    allowDeleting: true,
    allowTaskbarEditing: true,
    showDeleteConfirmDialog: true
  },
  toolbar: ['CriticalPath'],
  queryTaskbarInfo(args) {
    if (args.data.isCritical && !args.data.hasChildRecords) {
      args.taskbarBgColor = 'rgb(242, 210, 189)';
      args.progressBarBgColor = 'rgb(201, 169, 166)';
    }
  }
});
ganttChart.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

```

```

<style>
.e-gantt .e-gantt-chart .e-custom-holiday {
    background-color:#e82869;
}
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Rows in ##Platform_Name## Gantt control

Row represents a task information from the data source, and it is possible to perform the following actions in Gantt rows.

Row height

It is possible to change the height of the row in Gantt by setting row height in pixels to the [rowHeight](#) property. The following code example explains how to change the row height in Gantt at load time.

INDEX.JS

```

var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    rowHeight: 60
});
ganttChart.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Expand/Collapse Row

In Gantt parent tasks are expanded/collapsed by using expand/collapse icons, expand all/collapse all toolbar items and by using public methods. By default all tasks in Gantt was rendered in expanded state but we can change this status in Gantt.

Collapse all tasks at Gantt load

All tasks available in Gantt was rendered in collapsed state by setting [collapseAllParentTasks](#) property as `true`. The following code example shows how to use this property.

INDEX.JS

```

var ganttChart = new ej.gantt.Gantt({
  dataSource: GanttData,
  height: '450px',
  collapseAllParentTasks: true,
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  }
});

```

```
ganttChart.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Define expand/collapse status of tasks

In Gantt, we can render some tasks in collapsed state and some tasks in expanded state, this can done by defining expand status of the task in data source. This value was mapped to Gantt control by using [expandState](#) property. The following code example shows how to use this property.

INDEX.JS

```
var GanttData = [
  {
    TaskID: 1,
    TaskName: 'Project Initiation',
    StartDate: new Date('04/02/2019'),
    EndDate: new Date('04/21/2019'),
    isExpand: true,
    subtasks: [
      { TaskID: 2, TaskName: 'Identify Site location', StartDate:
new Date('04/02/2019'), Duration: 4, Progress: 50 },
      { TaskID: 3, TaskName: 'Perform Soil test', StartDate: new
Date('04/02/2019'), Duration: 4, Progress: 50 },
```



```

        { TaskID: 4, TaskName: 'Soil test approval', StartDate: new
Date('04/02/2019'), Duration: 4, Progress: 50 },
    ]
    },
    {
        TaskID: 5,
        TaskName: 'Project Estimation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        isExpand: false,
        subtasks: [
            { TaskID: 6, TaskName: 'Develop floor plan for estimation',
StartDate: new Date('04/04/2019'), Duration: 3, Progress: 50 },
            { TaskID: 7, TaskName: 'List materials', StartDate: new
Date('04/04/2019'), Duration: 3, Progress: 50 },
            { TaskID: 8, TaskName: 'Estimation approval', StartDate: new
Date('04/04/2019'), Duration: 3, Progress: 50 }
        ]
    },
];

var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        expandState: 'isExpand',
        child: 'subtasks'
    }
});
ganttChart.appendTo('#Gantt');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
```

```

    <div id="container">
        <div id="Gantt"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customize expand/collapse action

On expand action [expanding](#) and [expanded](#) event will be triggered with current expanding row's information. Similarly on collapse action [collapsing](#) and [collapsed](#) event will be triggered. Using this events and it's arguments we can customize the expand/collapse action. The following code example shows how to prevent the particular row from expand/collapse action using [expanding](#) and [collapsing](#) event.

INDEX.JS

```

var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    collapsing: function(args) {
        if (args.data.TaskID==1)
            args.cancel=true;
    },
    expanding: function(args) {
        if (args.data.TaskID==5)
            args.cancel=true;
    }
});
ganttChart.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Drag and drop

You can dynamically rearrange the rows in the Gantt control by using the `allowRowDragAndDrop` property. Using this property, row drag and drop can be enabled or disabled in Gantt. Using this feature, rows can be dropped at above and below as a sibling or child to the existing rows

To use row drag and drop feature, inject the `RowDD` and `Edit` module in Gantt.

INDEX.JS

```

var ganttChart = new ej.gantt.Gantt({
    dataSource: projectNewData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        dependency: 'Predecessor',
        child: 'subtasks'
    },
    allowRowDragAndDrop: true
});
ganttChart.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">

```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Gantt Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Multiple row drag and drop

Gantt also supports dragging multiple rows at a time and drop them on any rows above, below, or at child positions. In Gantt, you can enable the multiple drag and drop by setting the `selectionSettings.type` to `Multiple` and you should enable the `allowRowDragAndDrop` property.

INDEX.JS

```

var ganttChart = new ej.gantt.Gantt({
    dataSource: projectNewData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        dependency: 'Predecessor',
        child: 'subtasks'
    },
    selectionSettings: {
        type: 'Multiple'
    },
    allowRowDragAndDrop: true
});
ganttChart.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Taskbar drag and drop between rows

The Gantt feature empowers users to efficiently reorganize records by seamlessly moving taskbar and rearranging their positions through a simple drag-and-drop action. Using this feature, rows can be dropped at above and below as a sibling or child to the existing rows.

This mode can be enable by setting the [allowTaskbarDragAndDrop](#) property to `true`.

To use row drag and drop feature, inject the `RowDD` and `Edit` module in Gantt.

INDEX.JS

```

var ganttChart = new ej.gantt.Gantt({
  dataSource: projectNewData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    dependency: 'Predecessor',
    child: 'subtasks'
  }
});

```

```

    },
    editSettings: {
        allowAdding: true,
        allowEditing: true,
        allowDeleting: true,
        allowTaskbarEditing: true,
        showDeleteConfirmDialog: true
    },
    allowTaskbarDragAndDrop: true,
    allowRowDragAndDrop: true
});
gantChart.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Drag and drop events

We provide various events to customize the row drag and drop action, the following table explains about the available events and its details.

Event Name | Description

rowDragStartHelper | Triggers when clicking the drag icon or Gantt row.

rowDragStart |Triggers when drag action starts in Gantt.

rowDrag |Triggers while dragging the Gantt row.

rowDrop |Triggers when a drag row was dropped on the target row.

Customize row drag and drop action

In Gantt, the **rowDragStartHelper** and **rowDrop** events are triggered on row drag and drop action. Using this event, you can prevent dragging of particular record, validate the drop position, and cancel the drop action based on the target record and dragged record. The following topics explains about this.

Prevent dragging of particular record

You can prevent drag action of the particular record by setting the **cancel** property to **true**, which is available in the **rowDragStartHelper** event argument based on our requirement. In the following sample, drag action was restricted for first parent record and its child records.

INDEX.JS

```
var ganttChart = new ej.gantt.Gantt({
  dataSource: projectNewData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    dependency: 'Predecessor',
    child: 'subtasks'
  },
  rowDragStartHelper: function(args){
    var record = args.data[0] ? args.data[0] : args.data;
    var taskId = record.ganttProperties.taskId;
    if (taskId <= 4) {
      args.cancel = true;
    }
  },
  allowRowDragAndDrop: true
});
ganttChart.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
```

```

<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Validating drop position

You can prevent drop action based on the drop position and target record, by this, you can prevent dropping particular task on a specific task or specific position. This can be achieved by setting the `cancel` property to `true`, which is available in the `rowDrop` event argument.

In the following sample, we have prevented the drop action based on the position. In this sample, you cannot drop row as child in any of the available rows.

INDEX.JS

```

var ganttChart = new ej.gantt.Gantt({
    dataSource: projectNewData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        dependency: 'Predecessor',
        child: 'subtasks'
    },
    rowDrop: function(args) {
        if (args.dropPosition == "middleSegment") {
            args.cancel = true;
        }
    },
    allowRowDragAndDrop: true
});
ganttChart.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">

```



```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Gantt Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Prevent reordering a row as child to another row

You can prevent the default behavior of dropping rows as children to the target by setting the **cancel** property to **true** in [rowDrop](#) event argument. You can also change the drop position after cancelling using [reorderRows](#) method.

In the below example drop action is cancelled and dropped above to target row.

INDEX.JS

```

var ganttChart = new ej.gantt.Gantt({
    dataSource: projectNewData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        dependency: 'Predecessor',
        child: 'subtasks'
    },
    allowRowDragAndDrop: true,
    rowDrop : function (args) {
        if (args.dropPosition == 'middleSegment') {
            args.cancel = true;
            ganttChart.reorderRows([args.fromIndex], args.dropIndex,
'above');

```

```

    }
  }
});
gantChart.appendTo('#Gantt');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Perform row drag and drop action programmatically

Gantt provides option to perform row drag and drop action programmatically by using the `reorderRows` method, this method can be used for any external actions like button click. The following arguments are used to specify the positions to drag and drop a row:

- `fromIndexes`: Index value of source(dragging) row.
- `toIndex`: Value of target index.
- `position`: Drop positions such as above, below, or child.

The following code example shows how to drag and drop a row on button click action.

INDEX.JS

```

var ganttChart = new ej.gantt.Gantt({
  dataSource: projectNewData,
```

```

        height: '450px',
        taskFields: {
            id: 'TaskID',
            name: 'TaskName',
            startDate: 'StartDate',
            duration: 'Duration',
            progress: 'Progress',
            dependency: 'Predecessor',
            child: 'subtasks'
        },
        allowRowDragAndDrop: true
    });
    ganttChart.appendTo('#Gantt');
    var dragBtn= new ej.buttons.Button();
    dragBtn.appendTo('#dynamicDrag');
    document.getElementById('dynamicDrag').addEventListener('click', function()
    {
        ganttChart.reorderRows([1,2,3], 4, 'child');
    });

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="dynamicDrag">Drop records as child</button>
        <div id="Gantt"></div>
    </div>
    <script>
    var ele = document.getElementById('container');
    if(ele) {
        ele.style.visibility = "visible";
    }
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

Customize rows

You can customize the appearance of a row in grid side, by using the [rowDataBound](#) event and in chart side by using [queryTaskbarInfo](#) event

INDEX.JS

```
var ganttChart = new ej.gantt.Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  rowDataBound: function(args) {
    if (args.data['TaskID'] == 4) {
      args.row.style.background = 'cyan';
    }
  },
  queryTaskbarInfo: function(args) {
    if (args.data['TaskID'] == 4) {
      args.rowElement.style.background = 'cyan';
    }
  }
});
ganttChart.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
</body>
</html>
```

```

var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Styling alternate rows

You can change the background colour of alternative rows in Gantt chart, by overriding the class as shown below.

```

,
.e-altrow, tr.e-chart-row:nth-child(even) {
background-color: #f2f2f2;
}
,

```

INDEX.JS

```

ej.gantt.Gantt.Inject(ej.gantt.Toolbar);
var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
});
ganttChart.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <style>
        .e-altrow, tr.e-chart-row:nth-child(even) {
            background-color: #f2f2f2;
        }
    </style>

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Row spanning

Gantt chart has an option to span row cells. You can achieve this using [rowSpan](#) attribute to span cells in the [QueryCellInfo](#) event.

In the following demo, **Soil test approval** cell is spanned to two rows in the **TaskName** column.

INDEX.JS

```

var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    gridLines: 'Both',
    queryCellInfo: function(args) {
        if (args.data['TaskID'] == 4 && args.column.field ===
'TaskName') {
            args.rowSpan = 2;
        }
    },
});
ganttChart.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">

```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Gantt Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Scrolling in ##Platform_Name## Gantt control

The scrollbar will be displayed in the gantt when content exceeds the element **width** or **height**. The vertical and horizontal scrollbars will be displayed based on the following criteria:

- The vertical scrollbar appears when the total height of rows present in the gantt exceeds its element height.
- The horizontal scrollbar appears when the sum of columns width exceeds the grid pane size.
- The [height](#) and [width](#) are used to set the gantt height and width, respectively.

The default value for **height** and **width** is **auto**.

Set width and height

We can even set pixel values to width and height of gantt container using [width](#) and [height](#) properties.

INDEX.JS

```

var gantt = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '350px',
    width: '600px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',

```

```

        progress: 'Progress',
        child: 'subtasks',
    },
    editSettings: {
        allowAdding: true,
        allowEditing: true,
        allowDeleting: true,
        allowTaskbarEditing: true,
        showDeleteConfirmDialog: true
    },
});
gantt.appendTo('#Gantt');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Responsive with the parent container

Specify the [width](#) and [height](#) as **100%** to make the gantt element fill its parent container.

Setting the **height** to **100%** requires the gantt parent element to have explicit height. Also, the component will be responsive when the parent container is resized.

INDEX.JS


```

var gantt = new ej.gantt.Gantt({
  dataSource: GanttData,
  height: '100%',
  width: '100%',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks',
  },
  editSettings: {
    allowAdding: true,
    allowEditing: true,
    allowDeleting: true,
    allowTaskbarEditing: true,
    showDeleteConfirmDialog: true
  },
});
gantt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <style>
    .e-ganttresize {
      resize: both;
      overflow: auto;
      border: 1px solid red;
      padding: 10px;
      height: 300px;
      min-height: 250px;
      min-width: 250px;
    }
    .e-text{
      font-family: Helvetica, sans-serif;
      font-size: 14px;
    }
  </style>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>

```

```

<body>

    <p class="e-text"> The parent container can be resizable by dragging the
    bottom-right corner.</p>
    <div id="container" class="e-ganttresize">
        <div id="Gantt"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Scroll To Date method

In the Gantt control, When We use the [scrollToDate](#) method, it will scroll the timeline horizontally to the date that we specified in the method's argument.

The following code examples show how the scroll To Date method works in Gantt:

INDEX.JS

```

var gantt = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        dependency: 'Predecessor',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    splitterSettings: {
        position: "50%"
    },
    projectStartDate: new Date('04/01/2019'),
    projectEndDate: new Date('05/30/2019'),
});
gantt.appendTo('#Gantt');
var scrollBtn= new ej.buttons.Button();
scrollBtn.appendTo('#scroll');
document.getElementById('scroll').addEventListener('click', function() {
    gantt.scrollToDate('05/27/2019');
});

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

```

```

<meta name="description" content="Typescript Gantt Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <button id="scroll">Scroll To Date</button>
    <div id="container">
      <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Set the vertical scroll position

In the Gantt control, you can set the vertical scroller position dynamically by clicking the custom button using the [setScrollTop](#) method.

INDEX.JS

```

var ganttChart = new ej.gantt.Gantt({
  dataSource: ganttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  splitterSettings: {
    position: "50%"
  }
});
ganttChart.appendTo('#Gantt');
var scrollBtn= new ej.buttons.Button();
scrollBtn.appendTo('#scrollTop');
document.getElementById('scrollTop').addEventListener('click', function() {
  ganttChart.ganttChartModule.scrollObject.setScrollTop(300);
});

```

```
});
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <button id="scrollTop">Change Scroll Position</button>
  <div id="container">
    <div id="Gantt"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Managing Tasks

Managing tasks in ##Platform_Name## Gantt control

The Gantt component has options to dynamically insert, delete, and update tasks in the project. The primary key column is necessary to manage the tasks and perform CRUD operations in Gantt. To define the primary key, set the [columns.isPrimaryKey](#) property to `true` in the particular column.

To use CRUD, inject the [Edit](#) module into the Gantt control.

INDEX.JS

```
ej.gantt.Gantt.Inject(ej.gantt.Edit);
var ganttChart = new ej.gantt.Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
```

```

        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    editSettings: {
        allowEditing: true,
        mode: 'Dialog'
    }
});
gantChart.appendTo('#Gantt');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Read-only Gantt

In Gantt, all create, update, delete operations can be disabled by set `readOnly` property as `true`. The following sample demonstrates, render Gantt chart as read only.

INDEX.JS

```

ej.gantt.Gantt.Inject(ej.gantt.Edit,ej.gantt.Toolbar);
var ganttChart = new ej.gantt.Gantt({
  dataSource: GanttData,
```

```

        height: '450px',
        taskFields: {
            id: 'TaskID',
            name: 'TaskName',
            startDate: 'StartDate',
            duration: 'Duration',
            progress: 'Progress',
            child: 'subtasks'
        },
        toolbar:
[ 'Add', 'Cancel', 'CollapseAll', 'Delete', 'Edit', 'ExpandAll', 'NextTimeSpan', 'PrevTimeSpan', 'Search', 'Update', 'Indent', 'Outdent'],
        enableContextMenu: true,
        editSettings: {
            allowEditing: true,
            allowAdding: true,
            allowDeleting: true,
            allowTaskbarEditing: true
        },
        allowSorting: true,
        allowResizing: true,
        readOnly: true
    });
    ganttChart.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <style>
        .e-gantt-chart .e-preventEdit .e-right-resize-gripper,
        .e-gantt-chart .e-preventEdit .e-left-resize-gripper,
        .e-gantt-chart .e-preventEdit .e-progress-resize-gripper,
        .e-gantt-chart .e-preventEdit .e-left-connectorpoint-outer-div,
        .e-gantt-chart .e-preventEdit .e-right-connectorpoint-outer-div {
            display: none;
        }
    </style>
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```

```

<div id="container">
  <div id="Gantt"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Troubleshoot: Editing works only when primary key column is defined

Editing feature requires a primary key column for CRUD operations. While defining columns in Gantt using the [columns](#) property, it is mandatory that any one of the columns, must be a primary column. By default, the [id](#) column will be the primary key column. If [id](#) column is not defined, we need to enable [isPrimaryKey](#) for any one of the columns defined in the [columns](#) property.

Open new task dialog with default values

You can set default values when new task dialog opens using [actionBegin](#) event when `requestType` is `beforeOpenAddDialog`.

INDEX.JS

```

ej.gantt.Gantt.Inject(ej.gantt.Toolbar);
var ganttChart = new ej.gantt.Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  editSettings: {
    allowAdding: true
  },
  actionBegin : function (args) {
    if (args.requestType == 'beforeOpenAddDialog') {
      args.rowData.TaskName = 'Gantt';
      args.rowData.Progress = 70;
      args.rowData.ganttProperties.taskName = 'Gantt';
      args.rowData.ganttProperties.progress = 70;
    }
  },
  toolbar: ['Add']
});
ganttChart.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>

```

```

<title>EJ2 Gantt</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Gantt Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Adding new tasks in ##Platform_Name## Gantt control

Tasks can be dynamically added to the Gantt project by enabling the [editSettings.allowAdding](#) property.

Toolbar

A row can be added to the Gantt component from the toolbar while the [editSettings.allowAdding](#) property is set to true. On clicking the toolbar add icon, you should provide the task information in the add dialog.

INDEX.JS

```

ej.gantt.Gantt.Inject(ej.gantt.Toolbar);
var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    editSettings: {
        allowAdding: true
    },

```



```

        toolbar: ['Add']

    });
    ganttChart.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

By default, the new row will be added to the top most row in the Gantt control.

Context menu

A row can also be added above, below or child of the selected row by using context menu support. For this, we need to enable the property [enableContextMenu](#) and inject the [ContextMenu](#) module into the Gantt control.

INDEX.JS

```

var ganttChart = new ej.gantt.Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',

```

```

        progress: 'Progress',
        child: 'subtasks'
    },
    enableContextMenu: true,
    editSettings: {
        allowAdding: true
    },
    });
ganttChart.appendTo('#Gantt');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Using method

You can add rows to the Gantt control dynamically using the [addRecord](#) method and you can define the add position of the default new record by using the [rowPosition](#) property. You can also pass the `rowIndex` as an additional parameter.

- Top of all the rows.
- Bottom to all the existing rows.
- Above the selected row.
- Below the selected row.

- As child to the selected row.

INDEX.JS

```

ej.gantt.Gantt.Inject(ej.gantt.Toolbar,ej.gantt.Edit);
var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    editSettings:{
        allowAdding:true
    }
});
ganttChart.appendTo('#Gantt');
var addBtn= new ej.buttons.Button();
addBtn.appendTo('#addRow');
document.getElementById('addRow').addEventListener('click', function() {
    var record={ TaskID: 10,
        TaskName: 'New Added Record',
        StartDate: new Date('04/02/2019'),
        Duration: 3,
        Progress: 50
    };
    ganttChart.editModule.addRecord(record,'Below',2);
});

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">

```

```

        <button id="addRow">Add Row</button>
        <div id="Gantt"></div>
    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Editing tasks in ##Platform_Name## Gantt control

The editing feature can be enabled in the Gantt control by enabling the [editSettings.allowEditing](#) and [editSettings.allowTaskbarEditing](#) properties.

The following editing options are available to update the tasks in the Gantt chart:

- Cell
- Dialog
- Taskbar
- Dependency links

Cell editing

By setting the edit mode to auto using the [editSettings.mode](#) property, the tasks can be edited through TreeGrid cells by double-clicking.

Note: If the [Edit](#) module is not injected, you cannot edit the tasks through TreeGrid cells.

The following code example shows you how to enable the cell editing in Gantt control.

INDEX.JS

```

ej.gantt.Gantt.Inject(ej.gantt.Edit);
var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    editSettings: {
        allowEditing: true,
        mode: 'Auto'
    }
});
ganttChart.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Note: When the edit mode is set to **Auto**, on performing double-click action on TreeGrid side, the cells will be changed to editable mode and on performing double-click action on chart side, the edit dialog will appear for editing the task details.

Dialog editing

Modify the task details through the edit dialog by setting the edit [mode](#) to **Dialog**.

INDEX.JS

```

ej.gantt.Gantt.Inject(ej.gantt.Edit);
var ganttChart = new ej.gantt.Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  editSettings: {
    allowEditing: true,
    mode: 'Dialog'
  }
});

```

```

    }
    });
    ganttChart.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Note: In dialog editing mode, the edit dialog appears when performing double-click action on both TreeGrid or Gantt chart sides.

Sections or tabs in dialog

In the Gantt dialog, you can define the required tabs or editing sections using the [addDialogFields](#) and [editDialogFields](#) properties. Every tab is defined using the [type](#) property.

INDEX.JS

```

ej.gantt.Gantt.Inject(ej.gantt.Edit,ej.gantt.Toolbar);
var ProjectResources= [
  { resourceId: 1, resourceName: 'Martin Tamer' },
  { resourceId: 2, resourceName: 'Rose Fuller' },
  { resourceId: 3, resourceName: 'Margaret Buchanan' },
  { resourceId: 4, resourceName: 'Fuller King' },
  { resourceId: 5, resourceName: 'Davolio Fuller' },
  { resourceId: 6, resourceName: 'Van Jack' },

```

```

    { resourceId: 7, resourceName: 'Fuller Buchanan' },
    { resourceId: 8, resourceName: 'Jack Davolio' },
    { resourceId: 9, resourceName: 'Tamer Vinet' },
    { resourceId: 10, resourceName: 'Vinet Fuller' },
    { resourceId: 11, resourceName: 'Bergs Anton' },
    { resourceId: 12, resourceName: 'Construction Supervisor' }
];

var GanttData = [
    {
        TaskID: 1,
        TaskName: 'Project Initiation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        isParent:true,
        subtasks: [
            { TaskID: 2, TaskName: 'Identify Site location', StartDate:
new Date('04/02/2019'), Duration: 0, Progress: 50,isParent:false,info:
'Measure the total property area allotted for construction' },
            { TaskID: 3, TaskName: 'Perform Soil test', StartDate: new
Date('04/02/2019'), Duration: 4, Progress: 50, resources: [2, 3,
5],isParent:false,info: 'Obtain an engineered soil test of lot where
construction is planned.' +
                'From an engineer or company specializing in soil
testing' },
            { TaskID: 4, TaskName: 'Soil test approval', StartDate: new
Date('04/02/2019'), Duration: 4,Predecessor:"2FS", Progress:
50,isParent:false },
        ]
    },
    {
        TaskID: 5,
        TaskName: 'Project Estimation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        isParent:true,
        subtasks: [
            { TaskID: 6, TaskName: 'Develop floor plan for estimation',
StartDate: new Date('04/04/2019'), Duration: 3, Progress: 50, resources:
[4],
                isParent:false,info: 'Develop floor plans and
obtain a materials list for estimations' },
            { TaskID: 7, TaskName: 'List materials', StartDate: new
Date('04/04/2019'), Duration: 3, Progress: 50, resources: [4,
8],isParent:false,info: '' },
            { TaskID: 8, TaskName: 'Estimation approval', StartDate: new
Date('04/04/2019'), Duration: 0,Predecessor:"6SS", Progress: 50, resources:
[12, 5],isParent:false,info: '' }
        ]
    },
];

var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height:'450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',

```

```

        resourceInfo: 'resources',
        duration: 'Duration',
        progress: 'Progress',
        notes: 'info',
        dependency: 'Predecessor',
        child: 'subtasks'
    },
    toolbar: ['Add', 'Edit', 'Delete', 'Cancel'],
    editDialogFields: [
        { type: 'General', headerText: 'General' },
        { type: 'Dependency' },
        { type: 'Resources' },
        { type: 'Notes' }
    ],
    addDialogFields: [
        { type: 'General', headerText: 'General' },
        { type: 'Dependency' }
    ],
    columns: [
        { field: 'TaskID', headerText: 'Task ID', width: '100' },
        { field: 'TaskName', headerText: 'Task Name', width: '250' },
        { field: 'isParent', headerText: 'Custom Column',
width: '200' },
        { field: 'resources', headerText: 'Resources', width: '200' },
        { field: 'StartDate', headerText: 'Start Date',
width: '150' },
        { field: 'Duration', headerText: 'Duration', width: '150' },
        { field: 'Progress', headerText: 'Progress', width: '150' },
    ],
    resourceFields: {
        id: 'resourceId',
        name: 'resourceName',
    },
    resources: ProjectResources,
    editSettings: {
        allowAdding: true,
        allowEditing: true,
        mode: 'Dialog',
        allowTaskbarEditing: true
    }
});
gantChart.appendTo('#Gantt');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">
```



```

<style>
    .e-gantt-chart .e-preventEdit .e-right-resize-gripper,
    .e-gantt-chart .e-preventEdit .e-left-resize-gripper,
    .e-gantt-chart .e-preventEdit .e-progress-resize-gripper,
    .e-gantt-chart .e-preventEdit .e-left-connectorpoint-outer-div,
    .e-gantt-chart .e-preventEdit .e-right-connectorpoint-outer-div {
        display: none;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Limiting data fields in general tab

In the Gantt dialog, you can make only specific data source fields visible for editing by using the [addDialogFields](#) and [editDialogFields](#) properties. The data fields are defined with [type](#) and [fields](#) properties.

Note: You can also define the custom fields in the add/edit dialog General tab using the [fields](#) property.

INDEX.JS

```

ej.gantt.Gantt.Inject(ej.gantt.Edit,ej.gantt.Toolbar);
var ProjectResources= [
    { resourceId: 1, resourceName: 'Martin Tamer' },
    { resourceId: 2, resourceName: 'Rose Fuller' },
    { resourceId: 3, resourceName: 'Margaret Buchanan' },
    { resourceId: 4, resourceName: 'Fuller King' },
    { resourceId: 5, resourceName: 'Davolio Fuller' },
    { resourceId: 6, resourceName: 'Van Jack' },
    { resourceId: 7, resourceName: 'Fuller Buchanan' },
    { resourceId: 8, resourceName: 'Jack Davolio' },
    { resourceId: 9, resourceName: 'Tamer Vinet' },
    { resourceId: 10, resourceName: 'Vinet Fuller' },
    { resourceId: 11, resourceName: 'Bergs Anton' },
    { resourceId: 12, resourceName: 'Construction Supervisor' }
];
var GanttData = [
    {

```

```

        TaskID: 1,
        TaskName: 'Project Initiation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        isParent:true,
        subtasks: [
            { TaskID: 2, TaskName: 'Identify Site location', StartDate:
new Date('04/02/2019'), Duration: 0, Progress: 50,isParent:false },
            { TaskID: 3, TaskName: 'Perform Soil test', StartDate: new
Date('04/02/2019'), Duration: 4, Progress: 50, resources: [2, 3,
5],isParent:false },
            { TaskID: 4, TaskName: 'Soil test approval', StartDate: new
Date('04/02/2019'), Duration: 4,Predecessor:"2FS", Progress:
50,isParent:false },
        ]
    },
    {
        TaskID: 5,
        TaskName: 'Project Estimation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        isParent:true,
        subtasks: [
            { TaskID: 6, TaskName: 'Develop floor plan for estimation',
StartDate: new Date('04/04/2019'), Duration: 3, Progress: 50, resources:
[4],isParent:false },
            { TaskID: 7, TaskName: 'List materials', StartDate: new
Date('04/04/2019'), Duration: 3, Progress: 50, resources: [4,
8],isParent:false },
            { TaskID: 8, TaskName: 'Estimation approval', StartDate: new
Date('04/04/2019'), Duration: 0,Predecessor:"6SS", Progress: 50, resources:
[12, 5],isParent:false }
        ]
    },
];
var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height:'450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        resourceInfo: 'resources',
        duration: 'Duration',
        progress: 'Progress',
        dependency: 'Predecessor',
        child: 'subtasks'
    },
    toolbar: ['Add','Edit','Delete','Cancel'],
    editDialogFields: [
        { type: 'General', headerText: 'General',fields:['TaskID',
'TaskName','isParent']},
        { type: 'Dependency' },
        { type: 'Resources' }
    ],
    addDialogFields: [

```

```

        { type: 'General', headerText: 'General', fields: ['TaskID',
'TaskName', 'isParent'] },
        { type: 'Resources' },
        { type: 'Dependency' }
    ],
    columns: [
        { field: 'TaskID', headerText: 'Task ID', width: '100' },
        { field: 'TaskName', headerText: 'Task Name', width: '250' },
        { field: 'isParent', headerText: 'Custom Column',
width: '100' },
        { field: 'resources', headerText: 'Resources', width: '200' },
        { field: 'StartDate', headerText: 'Start Date',
width: '150' },
        { field: 'Duration', headerText: 'Duration', width: '150' },
        { field: 'Progress', headerText: 'Progress', width: '150' },
    ],
    resourceFields: {
        id: 'resourceId',
        name: 'resourceName',
    },
    resources: ProjectResources,
    editSettings: {
        allowAdding: true,
        allowEditing: true,
        mode: 'Dialog',
        allowTaskbarEditing: true
    }
});
gantChart.appendTo('#Gantt');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <style>
    .e-gantt-chart .e-preventEdit .e-right-resize-gripper,
    .e-gantt-chart .e-preventEdit .e-left-resize-gripper,
    .e-gantt-chart .e-preventEdit .e-progress-resize-gripper,
    .e-gantt-chart .e-preventEdit .e-left-connectorpoint-outer-div,
    .e-gantt-chart .e-preventEdit .e-right-connectorpoint-outer-div {
      display: none;
    }
  </style>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Validation in ##Platform_Name## Gantt control

Column validation

Column validation validates the editing and adding data and it display errors for invalid fields before saving data. This is effective in both inline and dialog editing.

Gantt uses [Form Validator](#) component for column validation. You can set [validation rules](#) by defining the [columns.validationRules](#). The value cannot be saved unless the validation rule get satisfied.

INDEX.JS

```

ej.gantt.Gantt.Inject(ej.gantt.Edit,ej.gantt.Selection,ej.gantt.Toolbar);
var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    toolbar: ['Add', 'Edit', 'Update', 'Delete', 'Cancel'],
    columns: [
        { field: 'TaskID' },
        { field: 'TaskName', validationRules: { required: true } },
        { field: 'StartDate',editType: 'datetimepickeredit', edit: {
params: { format: 'M/d/y hh:mm a' } },
        format: { format: 'M/d/y hh:mm a', type: 'dateTime' } },
validationRules: { required: true, date: true } },
        { field: 'Duration', validationRules: { required: true } },
        { field: 'Progress', validationRules: { required: true } }
    ],
    editSettings: {
        allowAdding: true,
        allowEditing: true,
        allowDeleting: true,
        allowTaskbarEditing: true,

```

```

        showDeleteConfirmDialog: true
    }
});
gantChart.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Custom validation

You can define your own custom validation rules for the specific columns by using callback function to it's [validation rule](#).

In the below demo, custom validation applied for **TaskName** column.

INDEX.JS

```

ej.gantt.Gantt.Inject(ej.gantt.Edit,ej.gantt.Selection,ej.gantt.Toolbar);
var customFn = function(args) {
  return args['value'].length >= 8;
};
var ganttChart = new ej.gantt.Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',

```

```

        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    toolbar: ['Add', 'Edit', 'Update', 'Delete', 'Cancel'],
    columns: [
        { field: 'TaskID' },
        { field: 'TaskName', validationRules: { required: true,
minLength: [customFn, 'Value should be greater than 8 letters'] } },
        { field: 'StartDate', editType: 'datetimepickeredit', edit: {
params: { format: 'M/d/y hh:mm a' } },
        format: { format: 'M/d/y hh:mm a', type: 'dateTime' },
validationRules: { required: true, date: true } },
        { field: 'Duration', validationRules: { required: true } },
        { field: 'Progress', validationRules: { required: true } }
    ],
    editSettings: {
        allowAdding: true,
        allowEditing: true,
        allowDeleting: true,
        allowTaskbarEditing: true,
        showDeleteConfirmDialog: true
    }
});
ganttChart.appendTo('#Gantt');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
```

```

    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Dependency and resource grid validation

Validation rules can also be implemented for the dependency and resource grid in the add or edit dialog by employing the event [actionBegin](#).

Within the actionBegin event, validationRules can be configured for columns in the grid of the dependency and resource tabs using the requestType `beforeOpenEditDialog` or `beforeOpenAddDialog`.

INDEX.JS

```

ej.gantt.Gantt.Inject(ej.gantt.Edit,ej.gantt.Selection,ej.gantt.Toolbar);
var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        dependency: 'Predecessor',
        progress: 'Progress',
        child: 'subtasks',
        resourceInfo: 'resources'
    },
    resourceFields: {
        id: 'resourceId',
        name: 'resourceName'
    },
    resources: editingResources,
    actionBegin: function(args) {
        if (args.requestType == "beforeOpenEditDialog" ||
args.requestType == "beforeOpenAddDialog") {
            args.Dependency.columns[3].validationRules = { required:
true }

            args.Resources.columns[2].allowEditing = true
            args.Resources.columns[2].validationRules = { required: true
}
        }
    },
    toolbar: ['Add', 'Edit', 'Update', 'Delete', 'Cancel'],
    columns: [
        { field: 'TaskID' },
        { field: 'TaskName', validationRules: { required: true } },
        { field: 'StartDate', editType: 'datetimepickeredit', edit: {
params: { format: 'M/d/y hh:mm a' } },
format: { format: 'M/d/y hh:mm a', type: 'dateTime' },
validationRules: { required: true, date: true } },
        { field: 'Duration', validationRules: { required: true } },
        { field: 'Progress', validationRules: { required: true } }
    ],

```

```

        editSettings: {
            allowAdding: true,
            allowEditing: true,
            allowDeleting: true,
            allowTaskbarEditing: true,
            showDeleteConfirmDialog: true
        }
    });
    ganttChart.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Deleting tasks in ##Platform_Name## Gantt control

A task delete option in the Gantt control can be enabled by enabling the [editSettings.allowDeleting](#) property. Tasks can be deleted by clicking the delete toolbar item or using the `deleteRow` method. You can call this method dynamically on any custom actions like button click. The following code example shows how to enable the delete option in the Gantt control.

INDEX.JS

```

ej.gantt.Gantt.Inject(ej.gantt.Edit);
var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,

```



```

        height: '450px',
        taskFields: {
            id: 'TaskID',
            name: 'TaskName',
            startDate: 'StartDate',
            duration: 'Duration',
            progress: 'Progress',
            child: 'subtasks'
        },
        editSettings: {
            allowDeleting: true
        }
    });
    ganttChart.appendTo('#Gantt');
    var delBtn= new ej.buttons.Button();
    delBtn.appendTo('#deleteRecord');
    document.getElementById('deleteRecord').addEventListener('click', () => {
        ganttChart.editModule.deleteRow();
    });

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <button id="deleteRecord">Delete Record</button>

    <div id="container">
        <div id="Gantt"></div>
    </div>
    <script>
    var ele = document.getElementById('container');
    if(ele) {
        ele.style.visibility = "visible";
    }
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

NOTE: You should select any one of the rows in the Gantt control to perform task delete action.

You should set the [allowDeleting](#) value to `true` to delete the record dynamically.

Delete confirmation message

Delete confirmation message is used to get the confirmation from users before deleting a task. This confirmation message can be enabled by setting the [editSettings.showDeleteConfirmDialog](#) property to `true`.

The following code snippet explains how to enable the delete confirmation message in Gantt.

INDEX.JS

```
ej.gantt.Gantt.Inject(ej.gantt.Edit,ej.gantt.Toolbar);
var ganttChart = new ej.gantt.Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  toolbar: ['Delete'],
  editSettings: {
    allowDeleting: true,
    showDeleteConfirmDialog: true
  }
});
ganttChart.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <style>
    .e-gantt-chart .e-preventEdit .e-right-resize-gripper,
    .e-gantt-chart .e-preventEdit .e-left-resize-gripper,
    .e-gantt-chart .e-preventEdit .e-progress-resize-gripper,
    .e-gantt-chart .e-preventEdit .e-left-connectorpoint-outer-div,
    .e-gantt-chart .e-preventEdit .e-right-connectorpoint-outer-div {
      display: none;
    }
  </style>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Task bar editing in ##Platform_Name## Gantt control

Modify the task details through user interaction such as resizing and dragging the taskbar by enabling the [allowTaskbarEditing](#) property.

INDEX.JS

```

ej.gantt.Gantt.Inject(ej.gantt.Edit);
var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    editSettings: {
        allowTaskbarEditing: true
    }
});
ganttChart.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet" type="text/css">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Prevent editing for specific tasks

On taskbar edit action, the [taskbarEditing](#) event will be triggered. You can prevent the taskbar from editing using the [taskbarEditing](#) event. This can be done by setting cancel property of [taskbarEditing](#) event argument to true. And we can hide the taskbar editing indicators like taskbar resizer, progress resizer and connector points by using [queryTaskbarInfo](#) event. The following code example shows how to achieve this.

INDEX.JS

```

ej.gantt.Gantt.Inject(ej.gantt.Edit);
var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    taskbarEditing: function (args) {
        if (args.data.TaskID == 4) // We can't edit Task Id 4
            args.cancel = true;
    },
    queryTaskbarInfo: function (args) {
        if (args.data.TaskID == 6) {
            args.taskbarElement.className += ' e-preventEdit' // Taskbar
            editing indicators are hidden
        }
    },
    editSettings: {
        allowTaskbarEditing: true
    }
})

```

```
});
gantChart.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <style>
    .e-gantt-chart .e-preventEdit .e-right-resize-gripper,
    .e-gantt-chart .e-preventEdit .e-left-resize-gripper,
    .e-gantt-chart .e-preventEdit .e-progress-resize-gripper,
    .e-gantt-chart .e-preventEdit .e-left-connectorpoint-outer-div,
    .e-gantt-chart .e-preventEdit .e-right-connectorpoint-outer-div {
      display: none;
    }
  </style>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Task dependencies

In the Gantt control, you can update the dependencies between the tasks and link the tasks interactively. The task dependencies can be mapped from the data source using the [dependency](#) property.

You can update the task dependencies using the following ways:

- Mouse interactions: Using connector points in the taskbar, you can perform drag and drop action to create task dependency links.
- Edit dialog: Create or remove the task dependencies using the **Dependency** tab in the edit dialog.
- Cell editing: Create or remove the task links using cell editing.

The following code example demonstrates how to enable task dependency editing in the Gantt chart using the [editSettings](#) property.

INDEX.JS

```
ej.gantt.Gantt.Inject(ej.gantt.Edit);
var GanttData = [
    {
        TaskID: 1,
        TaskName: 'Project Initiation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        isParent:true,
        subtasks: [
            { TaskID: 2, TaskName: 'Identify Site location', StartDate:
new Date('04/02/2019'), Duration: 0, Progress: 50 },
            { TaskID: 3, TaskName: 'Perform Soil test', StartDate: new
Date('04/02/2019'), Duration: 4, Progress: 50 },
            { TaskID: 4, TaskName: 'Soil test approval', StartDate: new
Date('04/02/2019'), Duration: 4,Predecessor:"2FS", Progress: 50 },
        ]
    },
    {
        TaskID: 5,
        TaskName: 'Project Estimation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        isParent:true,
        subtasks: [
            { TaskID: 6, TaskName: 'Develop floor plan for estimation',
StartDate: new Date('04/04/2019'), Duration: 3, Progress: 50 },
            { TaskID: 7, TaskName: 'List materials', StartDate: new
Date('04/04/2019'), Duration: 3, Progress: 50 },
            { TaskID: 8, TaskName: 'Estimation approval', StartDate: new
Date('04/04/2019'), Duration: 0,Predecessor:"6SS", Progress: 50 }
        ]
    },
];

var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height:'450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        dependency: 'Predecessor',
        child: 'subtasks'
    }
});
```

```

    },
    editSettings: {
      allowEditing: true,
      allowTaskbarEditing: true,
      mode: 'Auto'
    }
  });
  ganttChart.appendTo('#Gantt');

```

INDEX.HTML

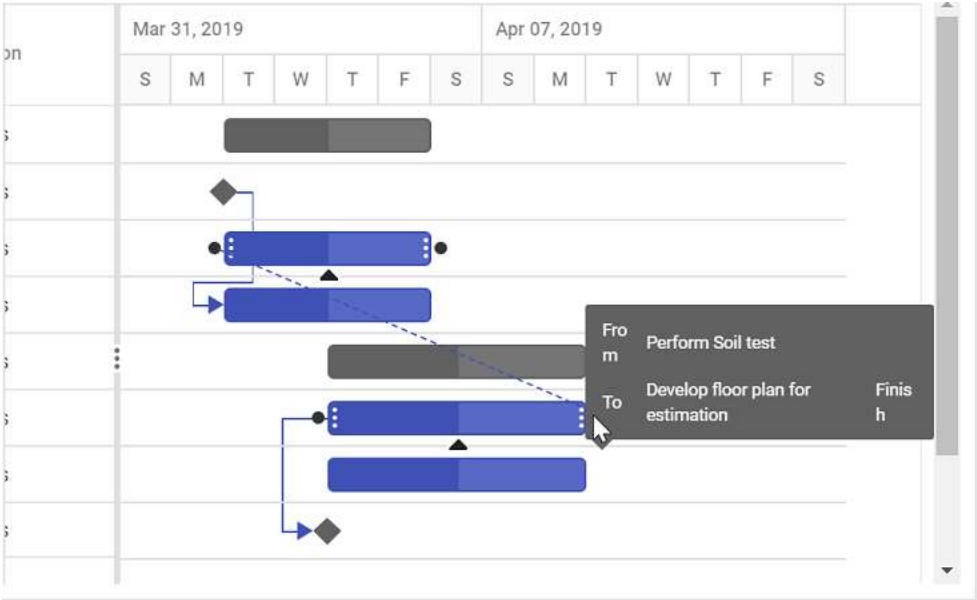
```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

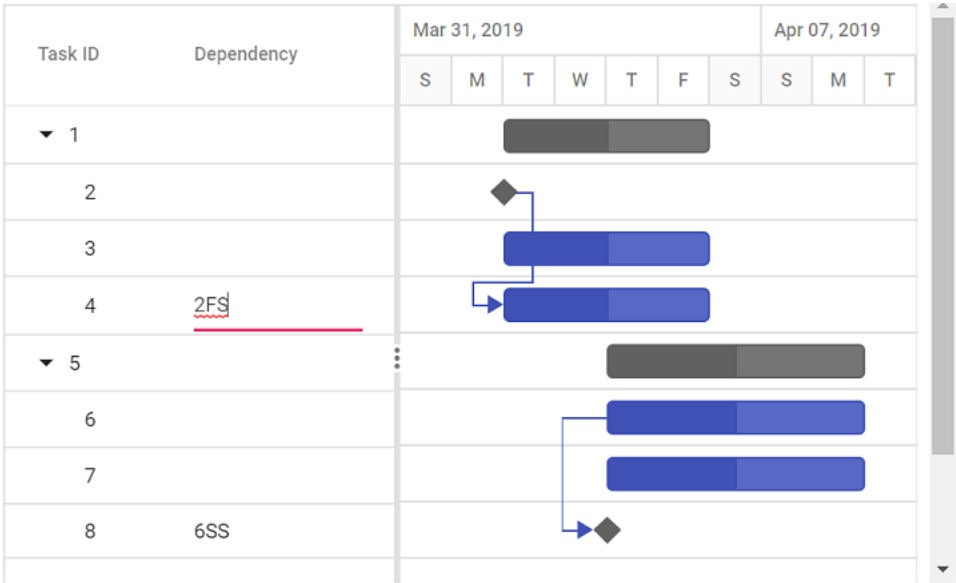
  <style>
    .e-gantt-chart .e-preventEdit .e-right-resize-gripper,
    .e-gantt-chart .e-preventEdit .e-left-resize-gripper,
    .e-gantt-chart .e-preventEdit .e-progress-resize-gripper,
    .e-gantt-chart .e-preventEdit .e-left-connectorpoint-outer-div,
    .e-gantt-chart .e-preventEdit .e-right-connectorpoint-outer-div {
      display: none;
    }
  </style>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```



Updating with mouse interaction action



Updating with cell edit

Task Information

×

GENERAL

DEPENDENCY

+

Add

🗑️

Delete

ID	Name	Type	Offset
2	2-Identify Site location	Finish-Start	0 days

SAVE

CANCEL

Updating with dialog

Note: When the edit mode is set to **Auto**, on performing double-click action on TreeGrid side, the cells will be changed to editable mode and on performing double-click action on chart side, the edit dialog will appear for editing the task details.

Update task values using method

Tasks' value can be dynamically updated by using the [updateRecordById](#) method. You can call this method on any custom action. The following code example shows how to use this method to update a task.

NOTE: Using the [updateRecordById](#) method, you cannot update the task ID value.

INDEX.JS

```

ej.gantt.Gantt.Inject(ej.gantt.Edit);
var ganttChart = new ej.gantt.Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  editSettings: {
    allowEditing: true
  }
});
ganttChart.appendTo('#Gantt');
var updateBtn= new ej.buttons.Button();
updateBtn.appendTo('#updateRecord');
document.getElementById('updateRecord').addEventListener('click', () => {
  var data={
    TaskID: 3,
    TaskName: 'Updated by index value',
  
```

```

        StartDate: new Date('04/02/2019'),
        Duration: 4,
        Progress: 50
    };
    ganttChart.updateRecordByID(data);
});

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <button id="updateRecord">Update Record</button>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Cell edit type and its params

The [columns.editType](#) is used to define the edit type for any particular column.

You can set the [columns.editType](#) based on data type of the column.

Also, you can customize the behavior of the editor component through the [columns.edit.params](#).

The following table describes cell edit type component and their corresponding edit params of the column.

Edit Type | Component | Example

INDEX.JS

```

ej.gantt.Gantt.Inject(ej.gantt.Toolbar);

```

```
function durationFormat(field, data, column) {
    return data[field];
}
var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    editSettings: {
        allowEditing: true
    },
    columns: [
        { field: 'TaskID', headerText: 'Task ID' },
        { field: 'TaskName', headerText: 'Task Name' },
        { field: 'StartDate', headerText: 'Start Date' },
        { field: 'Duration', headerText: 'Duration', editType: 'numericedit',
        edit: { params: { min: 1 } }, valueAccessor: durationFormat },
        { field: 'Progress', headerText: 'Progress', edit: { params: {
        showSpinButton: false } } }
    ],
});
ganttChart.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
```

```

if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Cell Edit Template

The cell edit template is used to create a custom component for a particular column by invoking the following functions:

- **create** - It is used to create the element at the time of initialization.
- **write** - It is used to create the custom component or assign default value at the time of editing.
- **read** - It is used to read the value from the component at the time of save.
- **destroy** - It is used to destroy the component.

INDEX.JS

```

var elem;
var dropdownlistObj;
var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    editSettings: {
        allowEditing: true
    },
    columns: [
        { field: 'TaskID', headerText: 'Task ID' },
        {
            field: 'TaskName', headerText: 'Task Name',
            edit: {
                create: function() {
                    elem = document.createElement('input');
                    return elem;
                },
                read: function() {
                    return dropdownlistObj.value;
                },
                destroy: function() {
                    dropdownlistObj.destroy();
                },
                write: function(args) {
                    dropdownlistObj = new ej.dropdowns.DropDownList({
                        dataSource: ganttChart.treeGrid.grid.dataSource,
                        fields: { value: 'TaskName' },
                        value: args.rowData[args.column.field],

```

```

        floatLabelType: 'Auto',
    });
    dropdownlistObj.appendTo(elem);
    }
    },
    { field: 'StartDate', headerText: 'Start Date' },
    { field: 'Duration', headerText: 'Duration' },
    { field: 'Progress', headerText: 'Progress' },
],
});
gantChart.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Disable editing for particular column

You can disable editing for particular columns, by using the [columns.allowEditing](#) property.

In the following demo, editing is disabled for the **TaskName** column.

INDEX.JS

```

var ganttChart = new ej.gantt.Gantt({
  dataSource: GanttData,

```

```

height: '450px',
taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
},
editSettings: {
    allowEditing: true
},
toolbar: ['Edit'],
columns: [
    { field: 'TaskID', headerText: 'Task ID' },
    { field: 'TaskName', headerText: 'Task Name', allowEditing: false },
    { field: 'StartDate', headerText: 'Start Date' },
    { field: 'Duration', headerText: 'Duration' },
    { field: 'Progress', headerText: 'Progress' },
],
});
gantChart.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

In dent and out dent in ##Platform_Name## Gantt control

Indent and Outdent of a task are used to update the level of task in the hierarchical order of the task. It can be performed by enabling the [editSettings.allowEditing](#) property.

Indent - Selected task can be indented to the level of task to the hierarchical order. It can be performed by using in-built context menu or toolbar items. It can also be invoked by using the `indent` method dynamically on any action like external button click. The following code example shows how to enable indent option in the Gantt chart.

Outdent - Selected task can be outdented to the level of task from the hierarchical order. It can be performed by using in-built context menu or toolbar items. It can also be invoked by using the `outdent` method dynamically on any action like external button click. The following code example shows how to enable outdent option in the Gantt chart.

INDEX.JS

```
ej.gantt.Gantt.Inject(ej.gantt.Toolbar,ej.gantt.Edit);
var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    toolbar: ['Indent', 'Outdent'],
    editSettings: {
        allowEditing: true
    }
});
var ind= new ej.buttons.Button();
ind.appendTo('#indent');
var out= new ej.buttons.Button();
out.appendTo('#outdent');
document.getElementById('indent').addEventListener('click', function() {
    ganttChart.indent();
});
document.getElementById('outdent').addEventListener('click', function() {
    ganttChart.outdent();
});
ganttChart.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
```

```

<link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <button id="indent">Indent</button>
    <button id="outdent">Outdent</button>
    <div id="Gantt"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Splitting and merging tasks in ##Platform_Name## Gantt control

Splitting task at load time

To split task at load time, we can define segment details in both hierarchical and self-referential way.

Refer below link for more details.

- [Split task at load time](#)

Split task dynamically

The task can be splitted dynamically, either by using the context menu or dialog.

- **Dialog:** Segments tab is rendered in add/edit dialog, when the [taskFields.segments](#) or [taskFields.segmentId](#) property is mapped. Using this tab, we can split the task based on the original start and end date of a particular task.
- **Context menu:** When the [taskFields.segments](#) or [taskFields.segmentId](#) property is mapped and the [enableContextMenu](#) property is enabled, Split Task item will be included in the context menu.

INDEX.JS

```

ej.gantt.Gantt.Inject(ej.gantt.Edit, ej.gantt.Toolbar);
var ganttChart = new ej.gantt.Gantt({
  dataSource: GanttData,
  height: "450px",
  taskFields: {

```



```

        id: "TaskID",
        name: "TaskName",
        startDate: "StartDate",
        endDate: "EndDate",
        duration: "Duration",
        progress: "Progress",
        child: "subtasks",
        segments: "Segments"
    },
    toolbar: ['Add', 'Edit', 'Update', 'Delete', 'Cancel', 'ExpandAll',
'CollapseAll'],
    editSettings: {
        allowAdding: true,
        allowEditing: true,
        allowDeleting: true,
        allowTaskbarEditing: true,
        showDeleteConfirmDialog: true
    },
    enableContextMenu: true
});
gantChart.appendTo('#Gantt');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Merge tasks

The splitted tasks can be merged either by using the **Merge Task** item of the Context menu or by using the dialog. We can also merge the tasks, by simply dragging the segments together in the UI.

Limitations of Split tasks

1. Parent and milestone tasks cannot be split into segments.
2. The task must have a width greater than the timeline unit cell in order to be split.
3. Split task is not supported in the **Resource view**.

Maintaining data in server in ##Platform_Name## Gantt control

All the modified data in Gantt control can be maintained in the database using RESTful web services.

All the CRUD operations in the gantt are done through DataManager. The DataManager has an option to bind all the CRUD related data in server-side.

In the below section, we have explained how to get the edited data details on the server-side using the **UrlAdaptor**.

URL Adaptor

Please refer the [link](#) to create the **ADO.NET** Entity Data Model in Visual studio,

We can define data source for Gantt as instance of DataManager using **url** property of DataManager. Please Check the below code snippet to assign data source to Gantt.

```
`ts
import { Gantt } from '@syncfusion/ej2-gantt';
import { DataManager, UrlAdaptor } from '@syncfusion/ej2-data';
let dataSource: DataManager = new DataManager({
url: '/Home/UrlDatasource',
adaptor: new UrlAdaptor
});
let gantt: Gantt = new Gantt({
dataSource: dataSource,
height: '450px',
treeColumnIndex: 1,
taskFields: {
id: 'TaskId',
name: 'TaskName',
startDate: 'StartDate',
progress: 'Progress',
duration: 'Duration',
dependency: 'Predecessor',
```

```

child: 'SubTasks'
}
});
gantt.appendTo('#Gantt');
`ts
GanttDataSourceEntities db = new GanttDataSourceEntities();
public ActionResult UrlDatasource(DataManagerRequest dm)
{
List<GanttData>DataList = db.GanttDatas.ToList();
var count = DataList.Count();
return Json(new { result = DataList, count = count });
}
`

```

We can also do CRUD operations over Gantt data and save the changes to database. By using **BatchUrl** property of **DataManager**, we can communicate with the controller method to update the data source on CRUD operation. In gantt CRUD actions on task are dependent with other tasks. For example on editing the child record on chart side, corresponding parent item also will get affect and predecessor dependency task as well get affected. So in Gantt all the CRUD operations are considered to be batch editing where you will get all the affected records as collection. Please check the below code snippet to assign controller method to this property.

```

`ts
import { Gantt } from '@syncfusion/ej2-gantt';
import { DataManager, UrlAdaptor } from '@syncfusion/ej2-data';
let dataSource: DataManager = new DataManager({
url: '/Home/UrlDatasource',
adaptor: new UrlAdaptor,
batchUrl: "Home/BatchSave"
});
let gantt: Gantt = new Gantt({
dataSource: dataSource,
height: '450px',
treeColumnIndex: 1,
taskFields: {
id: 'TaskId',

```

```

name: 'TaskName',
startDate: 'StartDate',
progress: 'Progress',
duration: 'Duration',
dependency: 'Predecessor',
child: 'SubTasks'
}
});
ganttt.appendTo('#Gantt');
`ts
public class ICRUDModel<T> where T : class
{
public object key { get; set; }
public T value { get; set; }
public List<T> added { get; set; }
public List<T> changed { get; set; }
public List<T> deleted { get; set; }
}
GanttDataSourceEntities db = new GanttDataSourceEntities();
public ActionResult BatchSave([FromBody]ICRUDModel<GanttData> data)
{
List<GanttData> uAdded = new List<GanttData>();
List<GanttData> uChanged = new List<GanttData>();
List<GanttData> uDeleted = new List<GanttData>();
//...
return Json(new { addedRecords = uAdded, changedRecords = uChanged, deletedRecords = uDeleted });
}
`

```

This server method will be triggered for all the CRUD operations like adding, editing and deleting actions. We can handle those each operations separately inside this method with corresponding data received in this method argument. Also, when using the `UrlAdaptor`, you need to return the data as JSON from the controller action and the JSON object must contain a property as result with dataSource as its value and one more property count with the dataSource total records count as its value.

Insert action

Using the `added` argument of the `BatchUrl` method we can insert the newly added row to database and return the same to client side. please find the below code example for details.

```
`ts
GanttDataSourceEntities db = new GanttDataSourceEntities();
public ActionResult BatchSave([FromBody]ICRUDModel<GanttData> data)
{
    List<GanttData> uAdded = new List<GanttData>();
    //Performing insert operation
    if (data.added != null && data.added.Count() > 0)
    {
        foreach (var rec in data.added)
        {
            uAdded.Add(this.Create(rec));
        }
    }
    return Json(new { addedRecords = uAdded });
    //...
}

public GanttData Create(GanttData value)
{
    db.GanttDatas.Add(value);
    db.SaveChanges();
    return value;
}
`
```

Editing action

Using the `changed` argument of the `BatchUrl` method we can update the modified records to database and return the same to client side. please find the below code example for details.

```
`ts
GanttDataSourceEntities db = new GanttDataSourceEntities();
public ActionResult BatchSave([FromBody]ICRUDModel<GanttData> data)
{
    List<GanttData> uChanged = new List<GanttData>();
```

```
//Performing update operation
if (data.changed != null && data.changed.Count() > 0)
{
    foreach (var rec in data.changed)
    {
        uChanged.Add(this.Edit(rec));
    }
}
return Json(new { changedRecords = uChanged });
//...
}

public GanttData Edit(GanttData value)
{
    GanttData result = db.GanttDatas.Where(currentData => currentData.TaskId ==
value.TaskId).FirstOrDefault();
    if (result != null)
    {
        result.TaskId = value.TaskId;
        result.TaskName = value.TaskName;
        result.StartDate = value.StartDate;
        result.EndDate = value.EndDate;
        result.Duration = value.Duration;
        result.Progress = value.Progress;
        result.Predecessor = value.Predecessor;
        db.SaveChanges();
        return result;
    }
    else
    {
        return null;
    }
}
```

Delete action

Using the `deleted` argument of the `BatchUrl` method we can remove the deleted records from database and return the same to client side. on deleting the record we need to remove its corresponding child records as well if it exist from the data base. please find the below code example for details.

```
`ts

GanttDataSourceEntities db = new GanttDataSourceEntities();

public ActionResult BatchSave([FromBody]ICRUDModel<GanttData> data)
{
    List<GanttData> uDeleted = new List<GanttData>();
    //Performing delete operation
    if (data.deleted != null && data.deleted.Count() > 0)
    {
        foreach (var rec in data.deleted)
        {
            uDeleted.Add(this.Delete(rec.TaskId));
        }
    }
    return Json(new { deletedRecords = uDeleted });
}

public GanttData Delete(string value)
{
    var result = db.GanttDatas.Where(currentData => currentData.TaskId == value).FirstOrDefault();
    db.GanttDatas.Remove(result);
    RemoveChildRecords(value);
    db.SaveChanges();
    return result;
}

public void RemoveChildRecords(string key)
{
    var childList = db.GanttDatas.Where(x => x.ParentId == key).ToList();
    foreach (var item in childList)
    {
        db.GanttDatas.Remove(item);
    }
}
```

```

RemoveChildRecords(item.TaskId);
}
}
,

```

Columns

Columns in ##Platform_Name## Gantt control

The column displays information from a bound data source, and you can edit the values of column to update the task details through TreeGrid. The operations such as sorting, filtering, and searching can be performed based on column definitions. To display a Gantt column, the [field](#) property should be mapped from the data source to the column.

If the column [field](#) is not specified in the data source, the column values will be empty.

The [treeColumnIndex](#) property is used to define the expander column in the Gantt control to expand and collapse the child rows.

Defining columns

Using the [columns](#) property, you can define the columns in Gantt. If the columns are not defined, then the default columns will be rendered based on the mapped data source fields in the [taskFields](#) property. Refer to the following code example for defining the columns in Gantt along with their widths.

INDEX.JS

```

var GanttData = [
    {
        TaskID: 1,
        TaskName: 'Project Initiation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        subtasks: [
            { TaskID: 2, TaskName: 'Identify Site location', StartDate:
new Date('04/02/2019'), Duration: 4, Progress: 50 },
            { TaskID: 3, TaskName: 'Perform Soil test', StartDate: new
Date('04/02/2019'), Duration: 4, Progress: 50 },
            { TaskID: 4, TaskName: 'Soil test approval', StartDate: new
Date('04/02/2019'), Duration: 4, Progress: 50 },
        ]
    },
    {
        TaskID: 5,
        TaskName: 'Project Estimation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        subtasks: [
            { TaskID: 6, TaskName: 'Develop floor plan for estimation',
StartDate: new Date('04/04/2019'), Duration: 3, Progress: 50 },
            { TaskID: 7, TaskName: 'List materials', StartDate: new
Date('04/04/2019'), Duration: 3, Progress: 50 },
            { TaskID: 8, TaskName: 'Estimation approval', StartDate: new
Date('04/04/2019'), Duration: 3, Progress: 50 }
        ]
    },
];

```



```

var ganttChart = new ej.gantt.Gantt({
  dataSource: GanttData,
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks',
  },
  height: '450px',
  splitterSettings: {
    columnIndex: 2
  },
  columns: [
    { field: 'TaskID', width: '150' },
    { field: 'TaskName', width: '250' }
  ]
});
ganttChart.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Custom column header

The column header text can be defined using the [headerText](#) property, and you can customize the column headers using the [headerTemplate](#) property.

INDEX.JS

```
var ganttChart = new ej.gantt.Gantt({
  dataSource: GanttData,
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks',
  },
  height: '450px',
  splitterSettings: {
    columnIndex: 4
  },
  columns: [
    { field: 'TaskName', headerTemplate: '#projectName', width:
'150' },
    { field: 'StartDate', headerTemplate:
'#dateTemplate', width: '150' },
    { field: 'Duration', headerTemplate: '#durationTemplate',
headerText: 'Duration', width: '150' },
    { field: 'Progress', headerTemplate: '#progressTemplate',
headerText: 'Progress', width: '150' },
  ],
});
ganttChart.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
```

```

<div id="container">
  <div id="Gantt"></div>
</div>

  <script type="text/x-template" id="projectName">
    <div>
      <div>
         Task Name
      </div>
    </div>
  </script>
  <script type="text/x-template" id="dateTemplate">
    <div>
      <div>
         Start Date
      </div>
    </div>
  </script>
  <script type="text/x-template" id="durationTemplate">
    <div>
      <div>
         Duration
      </div>
    </div>
  </script>
  <script type="text/x-template" id="progressTemplate">
    <div>
      <div>
         Progress
      </div>
    </div>
  </script>
</script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Format

To format the cell values based on a specific culture, use the [columns.format](#) property. The Gantt control uses the [Internationalization](#) library to format **number** and **date** values.

INDEX.JS

```

var ganttChart = new ej.gantt.Gantt({
  dataSource: GanttData,
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',

```

```

        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    {
        rowHeight: 50,
        splitterSettings: {
            columnIndex: 3
        },
        height: '450px',
        columns: [
            { field: 'TaskID', headerText: 'Task ID', textAlign: 'Left',
width: '100' },
            { field: 'Progress', headerText: 'Progress', width:
'150', format: 'C' },
            { field: 'TaskName', headerText: 'Task Name', width: '150' },
            { field: 'StartDate', headerText: 'Start Date',
width: '150' },
            { field: 'Duration', headerText: 'Duration', width: '150' }
        ]
    }
]);
gantChart.appendTo('#Gantt');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

By default, the number and date values are formatted in **en-US** culture.

Number formatting

The number or integer values can be formatted using the following format strings.

Format | Description | Remarks

{ type:'date', format:'dd/MM/yyyy' } | 04/07/2019

{ type:'date', format:'dd.MM.yyyy' } | 04.07.2019

{ type:'date', skeleton:'short' } | 7/4/19

{ type: 'dateTime', format: 'dd/MM/yyyy hh:mm a' } | 04/07/2019 12:00 AM

{ type: 'dateTime', format: 'MM/dd/yyyy hh:mm:ss a' } | 07/04/2019 12:00:00 AM

INDEX.JS

```
var ganttChart = new ej.gantt.Gantt({
  dataSource: GanttData,
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  splitterSettings:{
    columnIndex:3
  },
  height:'450px',
  columns: [
    { field: 'TaskID', headerText: 'Task ID', textAlign: 'Left',
width: '100' },
    { field: 'TaskName', headerText: 'Task Name', width: '150' },
      { field: 'Progress', headerText: 'Progress', width:
'150',format: 'C' },
      { field: 'StartDate', headerText: 'Start Date',
width: '150', format: { format: 'dd/MM/yyyy', type: 'date' } },
      { field: 'Duration', headerText: 'Duration', width: '150' }
    ]
  });
ganttChart.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Column reordering in ##Platform_Name## Gantt control

The column reordering can be done by dragging a column header from one index to another index within the TreeGrid. To enable reordering, set the [allowReordering](#) property to true.

To use the column reordering feature, inject the **Reorder** module to the Gantt control.

INDEX.JS

```

ej.gantt.Gantt.Inject(ej.gantt.Reorder);
var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    allowReordering: true,
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks',
    },
    height: '450px',
    splitterSettings: {
        columnIndex: 5
    }
});
ganttChart.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

```

```

<meta name="description" content="Typescript Gantt Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

You can disable the reordering of a particular column by setting the [columns.allowReordering](#) property to `false`.

Reorder multiple columns

Multiple columns can be reordered at a time by using the [reorderColumns](#) method.

INDEX.JS

```

ej.gantt.Gantt.Inject(ej.gantt.Reorder);
var ganttChart = new ej.gantt.Gantt({
  dataSource: GanttData,
  allowReordering: true,
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  splitterSettings: {
    position: '100%'
  },
  height: '450px',
  columns: [
    { field: 'TaskID', headerText: 'Task ID', textAlign: 'Left',
width: '100' },
    { field: 'TaskName', headerText: 'Task Name', width: '150' },

```

```

        { field: 'StartDate', headerText: 'Start Date', width: '150'
    },
        { field: 'Duration', headerText: 'Duration', width: '150' },
        { field: 'Progress', headerText: 'Progress', width: '150' }
    ]

    });
ganttChart.appendTo('#Gantt');
var reorderMultipleCols = new ej.buttons.Button();
reorderMultipleCols.appendTo('#reorderMultipleCols');
document.getElementById('reorderMultipleCols').addEventListener('click',
function() {
    ganttChart.reorderColumns(['TaskID', 'TaskName'], 'Progress');
});

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="reorderMultipleCols">Reorder Task ID and Task Name to
Last</button>
        <div id="Gantt"></div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

Reorder Events

During the reorder action, the gantt component triggers the below three events.

1. The [columnDragStart](#) event triggers when column header element drag (move) starts.

2. The [columnDrag](#) event triggers when column header element is dragged (moved) continuously.
3. The [columnDrop](#) event triggers when a column header element is dropped on the target column.

INDEX.JS

```
ej.gantt.Gantt.Inject(ej.gantt.Reorder);
var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    allowReordering: true,
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    splitterSettings: {
        columnIndex: 4
    },
    height: '450px',
    columns: [
        { field: 'TaskID', headerText: 'Task ID', textAlign: 'Left',
width: '100' },
        { field: 'TaskName', headerText: 'Task Name', width: '150' },
        { field: 'StartDate', headerText: 'Start Date', width: '150' },
        { field: 'Duration', headerText: 'Duration', width: '150' },
        { field: 'Progress', headerText: 'Progress', width: '150' }
    ],
    columnDragStart: function() {
        alert('ResizeStart event is Triggered');
    },
    columnDrag: function() {
        alert('Resizing event is Triggered');
    },
    columnDrop: function() {
        alert('ResizeStop event is Triggered');
    }
});
ganttChart.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Column resizing in ##Platform_Name## Gantt control

The column width can be resized by clicking and dragging the right edge of the column header. While dragging, the width of the column will be resized immediately. Each column can be auto resized by double-clicking the right edge of the column header to fit the width of that column based on the widest cell content. To resize the column, set the [allowResizing](#) property to true. The following code example shows how to enable the column resize feature in the Gantt control.

To resize the column, inject the **Resize** module into the Gantt control.

INDEX.JS

```

ej.gantt.Gantt.Inject(ej.gantt.Resize);
var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    allowResizing: true,
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    splitterSettings:{
        columnIndex:4
    },
    height:'450px'
});
ganttChart.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
<title>EJ2 Gantt</title>

```

```

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Gantt Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

You can disable resizing for a particular column by setting the [columns.allowResizing](#) to `false`.

Defining minimum and maximum column width

The column resizing can be restricted between minimum and maximum widths by defining the [columns->minWidth](#) and [columns->maxWidth](#) properties.

In the following example, the minimum and maximum widths are defined for the `Duration`, and `Task Name` columns.

INDEX.JS

```

ej.gantt.Gantt.Inject(ej.gantt.Resize);
var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    allowResizing: true,
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    splitterSettings:{
        columnIndex:5
    },

```

```

        height: '450px',
        columns: [
            { field: 'TaskID', headerText: 'Task ID', textAlign: 'Left',
width: '100' },
            { field: 'TaskName', headerText: 'Task Name', width:
'200', minWidth: '150' ,maxWidth: '250' },
            { field: 'StartDate', headerText: 'Start Date',
width: '150' },
            { field: 'Duration', headerText: 'Duration', width:
'100', minWidth: '50' ,maxWidth: '200' },
            { field: 'Progress', headerText: 'Progress', width: '150' }
        ]
    });
    ganttChart.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

Column template in ##Platform_Name## Gantt control

A column template is used to customize the column's look. The following code example explains how to define the custom template in Gantt using the [template](#) property.

INDEX.JS

```

var ProjectResources= [
  { resourceId: 1, resourceName: 'Martin Tamer' },
  { resourceId: 2, resourceName: 'Rose Fuller' },
  { resourceId: 3, resourceName: 'Margaret Buchanan' },
  { resourceId: 4, resourceName: 'Fuller King' },
  { resourceId: 5, resourceName: 'Davolio Fuller' },
  { resourceId: 6, resourceName: 'Van Jack' },
  { resourceId: 7, resourceName: 'Fuller Buchanan' },
  { resourceId: 8, resourceName: 'Jack Davolio' },
  { resourceId: 9, resourceName: 'Tamer Vinet' },
  { resourceId: 10, resourceName: 'Vinet Fuller' },
  { resourceId: 11, resourceName: 'Bergs Anton' },
  { resourceId: 12, resourceName: 'Construction Supervisor' }
];
var GanttData = [
  {
    TaskID: 1,
    TaskName: 'Project initiation',
    StartDate: new Date('04/02/2019'),
    EndDate: new Date('04/21/2019'),
    subtasks: [
      {TaskID: 2, TaskName: 'Identify site location', StartDate: new
Date('04/02/2019'), Duration: 4,Progress: 50, resources: [1]},
      {TaskID: 3, TaskName: 'Perform soil test', StartDate: new
Date('04/02/2019'), Duration: 4, Predecessor: '2',Progress: 50, resources:
[2]},
      {TaskID: 4, TaskName: 'Soil test approval', StartDate: new
Date('04/02/2019'), Duration: 4, Predecessor: '3', Progress: 50 ,resources:
[3]},
    ]
  },
  {
    TaskID: 5,
    TaskName: 'Project estimation',
    StartDate: new Date('04/02/2019'),
    EndDate: new Date('04/21/2019'),
    subtasks: [
      {TaskID: 6, TaskName: 'Develop floor plan for estimation',
StartDate: new Date('04/04/2019'),Duration: 3, Predecessor: '4', Progress:
50, resources: [4]},
      {TaskID: 7, TaskName: 'List materials', StartDate: new
Date('04/04/2019'),Duration: 3, Predecessor: '6', resources: [3],Progress:
50},
      {TaskID: 8, TaskName: 'Estimation approval', StartDate: new
Date('04/04/2019'),Duration: 3, Predecessor: '7',
    }
    ]
  }
];
var ganttChart = new ej.gantt.Gantt({
  dataSource: GanttData,
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    resourceInfo: 'resources',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
  }
});

```

```

        child: 'subtasks',
    },
    rowHeight: 50,
    splitterSettings: {
        columnIndex: 3
    },
    height: '450px',
    columns: [
        { field: 'TaskID', headerText: 'Task ID', textAlign: 'Left',
width: '100' },
        { field: 'TaskName', headerText: 'Task Name', width: '150' },
            { field: 'resources', headerText: 'Resources', width:
'250', template: '#columnTemplate' },
            { field: 'StartDate', headerText: 'Start Date', width: '150' },
            { field: 'Duration', headerText: 'Duration', width: '150' },
            { field: 'Progress', headerText: 'Progress', width: '150' },
    ],
    resourceFields: {
        id: 'resourceId',
        name: 'resourceName',
    },
    resources: ProjectResources
});
gantChart.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <script type="text/x-jsrender" id="columnTemplate">
    ${if(ganttProperties.resourceNames) }

    <div class="image">
      <div style="display:inline-
block;width:100%;position:relative;left:30px;top:-
14px">${gantProperties.resourceNames}</div>
    </div>
    ${/if}
  </script>

```

```

    <div id="container">
      <div id="Gantt"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Column menu in ##Platform_Name## Gantt control

The column menu has options to integrate features like sorting, filtering, and autofit. It will show a menu with the integrated feature when users click the Multiple icon of the column. To enable the column menu, you should set the [showColumnMenu](#) property to true.

The default items are displayed in the following table:

Item	Description
----- -----	
SortAscending	Sort the current column in ascending order.
SortDescending	Sort the current column in descending order.
AutoFit	Auto fit the current column.
AutoFitAll	Auto fit all columns.
Filter	Show the filter option as given in the <code>filterSettings.type</code> property.

INDEX.JS

```

ej.gantt.Gantt.Inject(ej.gantt.Filter, ej.gantt.Sort, ej.gantt.Resize);
var GanttData = [
  {
    TaskID: 1,
    TaskName: 'Project initiation',
    StartDate: new Date('04/02/2019'),
    EndDate: new Date('04/21/2019'),
    subtasks: [
      {TaskID: 2, TaskName: 'Identify site location', StartDate: new
Date('04/02/2019'), Duration: 0, Progress: 50, resources: [1]},
      {TaskID: 3, TaskName: 'Perform soil test', StartDate: new
Date('04/02/2019'), Duration: 4, Predecessor: '2', Progress: 50, resources:
[2, 3, 5]},
      {TaskID: 4, TaskName: 'Soil test approval', StartDate: new
Date('04/02/2019'), Duration: 0, Predecessor: '3', Progress: 50 },
    ]
  },
  {
    TaskID: 5,
    TaskName: 'Project estimation',
    StartDate: new Date('04/02/2019'),

```

```

        EndDate: new Date('04/21/2019'),
        subtasks: [
            {TaskID: 6, TaskName: 'Develop floor plan for estimation',
StartDate: new Date('04/04/2019'),Duration: 3, Predecessor: '4', Progress:
50, resources: [4]},
            {TaskID: 7, TaskName: 'List materials', StartDate: new
Date('04/04/2019'),Duration: 3, Predecessor: '6', resources: [4,
8],Progress: 50},
            {TaskID: 8, TaskName: 'Estimation approval', StartDate: new
Date('04/04/2019'),Duration: 0, Predecessor: '7', resources: [12, 5]
            }
        ]
    ]]
var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    showColumnMenu:true,
    allowResizing: true,
    allowFiltering: true,
    allowSorting: true,
    splitterSettings:{
        position:'100%'
    },
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks',
    },
    height:'450px'
});
ganttChart.appendTo('#Gantt');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
```



```

<div id="container">
  <div id="Gantt"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

You can disable the column menu for a particular column by setting the `columns.showColumnMenu` to `false`.

Column menu events

During the resizing action, the gantt component triggers the below two events.

1. The [columnMenuOpen](#) event triggers before the column menu opens.
2. The [columnMenuClick](#) event triggers when the user clicks the column menu of the gantt.

INDEX.JS

```

ej.gantt.Gantt.Inject(ej.gantt.Filter, ej.gantt.Sort, ej.gantt.Resize);
var ganttChart = new ej.gantt.Gantt({
  dataSource: GanttData,
  height: '450px',
  showColumnMenu: true,
  allowFiltering: true,
  allowResizing: true,
  allowSorting: true,
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  splitterSettings: {
    position: '100%'
  },
  columnMenuOpen: function() {
    alert('columnMenuOpen event is Triggered');
  },
  columnMenuClick: function() {
    alert('columnMenuClick event is Triggered');
  },
  columns: [
    { field: 'TaskID', headerText: 'Task ID' },
    { field: 'Progress', headerText: 'Progress' },
    { field: 'TaskName', headerText: 'Task Name' },
    { field: 'StartDate', headerText: 'Start Date' },
    { field: 'Duration', headerText: 'Duration' }
  ],

```

```
});
ganttChart.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Custom Column Menu Item

Custom column menu items can be added by defining the [columnMenuItems](#). Actions for this customized items can be defined in the [columnMenuClick](#) event.

INDEX.JS

```
ej.gantt.Gantt.Inject(ej.gantt.Filter, ej.gantt.Sort, ej.gantt.Resize);
var ganttChart = new ej.gantt.Gantt({
  dataSource: GanttData,
  height: '450px',
  showColumnMenu: true,
  allowFiltering: true,
  allowResizing: true,
  allowSorting: true,
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
```

```

        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    columnMenuItems: [{text: 'Clear Sorting', id: 'ganttclearsorting'}],
    columnMenuClick: function(args) {
        if (args.item.id === 'ganttclearsorting') {
            ganttChart.clearSorting();
        }
    },
    splitterSettings: {
        position: '75%'
    },
    sortSettings: {
        columns: [{direction: "Ascending", field: "TaskID"}]
    },
    columns: [
        { field: 'TaskID', headerText: 'Task ID' },
        { field: 'Progress', headerText: 'Progress' },
        { field: 'TaskName', headerText: 'Task Name' },
        { field: 'StartDate', headerText: 'Start Date' },
        { field: 'Duration', headerText: 'Duration' }
    ],
});
ganttChart.appendTo('#Gantt');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if (ele) {
    ele.style.visibility = "visible";
}
}
```

```
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Customize menu items for particular columns

Sometimes, you have a scenario that to hide an item from column menu for particular columns. In that case, you need to define the [columnMenuOpenEventArgs.hide](#) as true in the [columnMenuOpen](#) event.

The following sample, **Filter** item was hidden in column menu when opens for the **Task Name** column.

INDEX.JS

```
ej.gantt.Gantt.Inject(ej.gantt.Filter, ej.gantt.Sort, ej.gantt.Resize);
var ganttChart = new ej.gantt.Gantt({
  dataSource: GanttData,
  height: '450px',
  showColumnMenu: true,
  allowFiltering: true,
  allowResizing: true,
  allowSorting: true,
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  splitterSettings: {
    position: '100%'
  },
  columns: [
    { field: 'TaskID', headerText: 'Task ID' },
    { field: 'Progress', headerText: 'Progress' },
    { field: 'TaskName', headerText: 'Task Name' },
    { field: 'StartDate', headerText: 'Start Date' },
    { field: 'Duration', headerText: 'Duration' }
  ],
  columnMenuOpen: function (args) {
    for (var item of args.items) {
      if (item.text === 'Filter' && args.column.field === 'TaskName')
      {
        item.hide = true;
      } else {
        item.hide = false;
      }
    }
  },
});
ganttChart.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Gantt Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Responsive columns in ##Platform_Name## Gantt control

You can toggle the column visibility based on media queries, which are defined in the [hideAtMedia](#). The [hideAtMedia](#) accepts valid [Media Queries](#).

INDEX.JS

```

var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    splitterSettings: {
        position: '75%'
    },
    height: '450px',
    columns: [
        { field: 'TaskID', headerText: 'Task ID', textAlign: 'Left',
width: '100' },
        { field: 'TaskName', headerText: 'Task Name', width:
'200',hideAtMedia: '(min-width: 700px)' },
        { field: 'StartDate', headerText: 'Start Date',
width: '150' },

```

```

        { field: 'Duration', headerText: 'Duration', width: '100',
hideAtMedia: '(max-width: 500px)' },
        { field: 'Progress', headerText: 'Progress', width: '150' }
    ]

});
gantChart.appendTo('#Gantt');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Change tree/expander column

The tree/expander column is a column in the Gantt control, that has icons to expand or collapse the parent records. You can define the tree column index in the Gantt control by using the [treeColumnIndex](#) property and the default value of this property is 0. The following code example shows how to use this property.

INDEX.JS

```

var ganttChart = new ej.gantt.Gantt({
  dataSource: GanttData,
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
```

```

                startDate: 'StartDate',
                duration: 'Duration',
                progress: 'Progress',
                child: 'subtasks'
            },
            splitterSettings: {
                columnIndex: 3
            },
            height: '450px',
            treeColumnIndex: 2
        });
        ganttChart.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Show or Hide columns dynamically

You can show or hide gantt columns dynamically using external buttons by invoking the [showColumn](#) or [hideColumn](#) method.

INDEX.JS

```

ej.gantt.Gantt.Inject(ej.gantt.Toolbar,ej.gantt.Edit);
var ganttChart = new ej.gantt.Gantt({

```

```

        dataSource: GanttData,
        height: '450px',
        taskFields: {
            id: 'TaskID',
            name: 'TaskName',
            startDate: 'StartDate',
            duration: 'Duration',
            progress: 'Progress',
            child: 'subtasks'
        },
        editSettings: {
            allowEditing: true
        },
        splitterSettings: {
            position: '75%'
        },
        columns: [
            { field: 'TaskID', headerText: 'Task ID' },
            { field: 'Progress', headerText: 'Progress' },
            { field: 'TaskName', headerText: 'Task Name' },
            { field: 'StartDate', headerText: 'Start Date' },
            { field: 'Duration', headerText: 'Duration' }
        ]
    });
    var show= new ej.buttons.Button();
    show.appendTo('#show');
    var hide= new ej.buttons.Button();
    hide.appendTo('#hide');
    document.getElementById('show').addEventListener('click', function() {
        ganttChart.showColumn(['TaskName', 'Duration']);
    });
    document.getElementById('hide').addEventListener('click', function() {
        ganttChart.hideColumn(['TaskName', 'Duration']);
    });
    ganttChart.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```



```

    <div id="container">
        <button id="show">Show</button>
        <button id="hide">Hide</button>
        <div id="Gantt"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Check box columns in ##Platform_Name## Gantt control

To render boolean values as checkbox in columns, you need to set [displayAsCheckBox](#) property as **true**.

INDEX.JS

```

ej.gantt.Gantt.Inject(ej.gantt.Toolbar,ej.gantt.Edit);
var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks',
        verified: 'verified'
    },
    splitterSettings: {
        columnIndex: 5
    },
    columns: [
        { field: 'TaskID', headerText: 'Task ID' },
        { field: 'Progress', headerText: 'Progress' },
        { field: 'TaskName', headerText: 'Task Name' },
        { field: 'StartDate', headerText: 'Start Date' },
        { field: 'verified', headerText: 'Verified', width: 150,
displayAsCheckBox: true, type: 'boolean' },
        { field: 'Duration', headerText: 'Duration' }
    ]
});
ganttChart.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">

```

```

<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Controlling Grid actions

You can enable or disable gantt action for a particular column by setting the [allowFiltering](#), [allowSorting](#), [allowReordering](#), and [allowEditing](#) properties.

INDEX.JS

```

ej.gantt.Gantt.Inject(ej.gantt.Toolbar,ej.gantt.Edit);
var ganttChart = new ej.gantt.Gantt({
  dataSource: GanttData,
  height: '450px',
  allowFiltering: true,
  allowSorting: true,
  allowReordering: true,
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks',
  },
  editSettings: {
    allowEditing: true
  },
  splitterSettings: {
    position: '75%'
  },
  columns: [
    { field: 'TaskID', headerText: 'Task ID' },

```

```

    { field: 'Progress', headerText: 'Progress', allowReordering: false
  },
    { field: 'TaskName', headerText: 'Task Name', allowSorting: false },
    { field: 'StartDate', headerText: 'Start Date', allowEditing: false
  },
    { field: 'Duration', headerText: 'Duration', allowFiltering: false }
  ]
});
gantChart.appendTo('#Gantt');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Column type

Column type can be specified using the `columns.type` property. It specifies the type of data the column binds.

If the `format` is defined for a column, the column uses `type` to select the appropriate format option [number](#) or [date](#).

Gantt column supports the following types:

- string

- number
- boolean
- date
- date time

If the `type` is not defined, it will be determined from the first record of the [dataSource](#).

In case if the first record of the [dataSource](#) is null/blank value for a column then it is necessary to define the `type` for that column.

Column spanning in ##Platform_Name## Gantt control

The gantt has option to span the adjacent cells. You need to define the [colSpan](#) attribute to span cells in the [QueryCellInfo](#) event.

In the following demo, **Work 1** cells have been spanned.

INDEX.JS

```
ej.gantt.Gantt.Inject(ej.gantt.Toolbar,ej.gantt.Edit);
var ganttChart = new ej.gantt.Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    work1: 'work1',
    work2: 'work2',
    progress: 'Progress',
    child: 'subtasks'
  },
  splitterSettings: {
    columnIndex: 5
  },
  gridLines: 'Both',
  columns: [
    { field: 'TaskID', headerText: 'Task ID' },
    { field: 'TaskName', headerText: 'Task Name' },
    { field: 'work1', headerText: 'Work 1' },
    { field: 'work2', headerText: 'Work 2' },
    { field: 'StartDate', headerText: 'Start Date' },
    { field: 'Duration', headerText: 'Duration' },
    { field: 'Progress', headerText: 'Progress' }
  ],
  queryCellInfo: function (args) {
    switch (args.data.TaskID) {
      case 1:
        if ((args.column.field == 'work1') &&
          (args.data.taskData.work1 == 'support')) {
          args.colSpan = 2;
        }
        break;
      case 2:
        if ((args.column.field == 'work1') &&
          (args.data.taskData.work1 == 'support')) {
```

```

        args.colSpan = 2;
    }
    break;
    case 3:
        if ((args.column.field == 'work1') &&
            (args.data.taskData.work1 == 'support')) {
            args.colSpan = 2;
        }
        break;
    case 4:
        if ((args.column.field == 'work1') &&
            (args.data.taskData.work1 == 'support')) {
            args.colSpan = 2;
        }
        break;
    case 5:
        if ((args.column.field == 'work1') &&
            (args.data.taskData.work1 == 'support')) {
            args.colSpan = 2;
        }
        break;
    case 7:
        if ((args.column.field == 'work1') &&
            (args.data.taskData.work1 == 'support')) {
            args.colSpan = 2;
        }
        break;
    }
}
});
gantChart.appendTo('#Gantt');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
```

```

    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

State persistence in ##Platform_Name## Gantt control

State persistence refers to the Gantt's state maintained in the browser's [localStorage](#) even if the browser is refreshed or if you move to the next page within the browser. State persistence stores gantt's model object in the local storage when the [enablePersistence](#) is defined as true.

Get or set localStorage value

If the [enablePersistence](#) property is set to true, the gantt property value is saved in the `window.localStorage` for reference. You can get/set the localStorage value by using the `getItem/setItem` method in the `window.localStorage`.

```
`ts
```

```
//get the Gantt model.
```

```
let value: string = window.localStorage.getItem('ganttGantt'); //"ganttGantt" is component name +
component id.
```

```
let model: Object = JSON.parse(model);
```

```
,
```

```
`ts
```

```
//set the Gantt model.
```

```
window.localStorage.setItem('ganttGantt', JSON.stringify(model)); //"ganttGantt" is component name +
component id.
```

```
,
```

Prevent columns from persisting

When the [enablePersistence](#) property is set to true, the Gantt properties such as [Filtering](#), [Sorting](#), and [Columns](#) will persist. You can use the `addOnPersist` method to prevent these Gantt properties from persisting.

The following example demonstrates how to prevent Gantt columns from persisting. In the [dataBound](#) event of the Gantt, you can override the `addOnPersist` method and remove the columns from the key list given for persistence.

Note: When the `enablePersistence` property is set to true, the Gantt features such as column template, column formatter, header text, and value accessor will not persist.

INDEX.JS

```

var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,

```

```

height: '450px',
taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks',
},
columns: [
    { field: 'TaskID', headerText: 'Task ID', textAlign: 'Right', width:
120 },
    { field: 'TaskName', headerText: 'Task Name', width: 150},
    { field: 'StartDate', headerText: 'StartDate', width: 150 },
    { field: 'Duration', headerText: 'Duration', width: 150},
],
enablePersistence: true,
editSettings: {
    allowAdding: true,
    allowEditing: true,
    allowDeleting: true,
    allowTaskbarEditing: true,
    showDeleteConfirmDialog: true
},
dataBound: dataBound
});
gantChart.appendTo('#Gantt');
function dataBound(args) {
    var cloned = this.addOnPersist;
    this.addOnPersist = function (key) {
        key = key.filter(function (item) { return item !== "columns"; });
        return cloned.call(this, key);
    };
}
document.getElementById('add').onclick = function () {
    var obj = { field: "Progress", headerText: 'Progress', width: 100 };
    gantChart.columns.push(obj);
    gantChart.treeGrid.refreshColumns();
};
document.getElementById('remove').onclick = function () {
    gantChart.columns.pop();
    gantChart.treeGrid.refreshColumns();
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="add">Add columns</button>
        <button id="remove">Remove columns</button>
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Persist the header template and header Text

By default, the Gantt properties such as column template, header text, header template, column formatter, and value accessor will not persist when [enablePersistence](#) is set to true. Because the column template and header text can be customized at the application level.

If you wish to restore all these column properties, then you can achieve it by cloning the gantt's columns property using JavaScript Object's assign method and storing this along with the persist data manually. While restoring the settings, this column object must be assigned to the gantt's columns property to restore the column settings as demonstrated in the following sample.

INDEX.JS

```

var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks',
    },
    columns: [
        { field: 'TaskID', headerText: 'Task ID', textAlign: 'Right', width:
120 },
        { field: 'TaskName', headerText: 'Task Name', width: 150,
headerTemplate: '#customertemplate' },
        { field: 'StartDate', headerText: 'StartDate', width: 150 },
        { field: 'Duration', headerText: 'Duration', width: 150},
    ]
});

```



```

        { field: 'Progress', headerText: 'Progress', width: 150 },
    ],
    enablePersistence: true,
    editSettings: {
        allowAdding: true,
        allowEditing: true,
        allowDeleting: true,
        allowTaskbarEditing: true,
        showDeleteConfirmDialog: true
    },
});
gantChart.appendTo('#Gantt');
var savedProperties;
document.getElementById('restore').onclick = function () {
    savedProperties = JSON.parse(gantChart.getPersistData());
    var gridColumnState = Object.assign([], gantChart.ganttColumns);
    savedProperties.columns.forEach(function (col) {
        var headerText = gridColumnState.find(function (colColumnsState) {
            return colColumnsState.field === col.field; })['headerText'];
        var colTemplate = gridColumnState.find(function (colColumnsState) {
            return colColumnsState.field === col.field; })['template'];
        var headerTemplate = gridColumnState.find(function
        (colColumnsState) { return colColumnsState.field === col.field;
        })['headerTemplate'];
        col.headerText = 'Text Changed';
        col.template = colTemplate;
        col.headerTemplate = headerTemplate;
    });
    console.log(savedProperties);
    gantChart.treeGrid.setProperties(savedProperties);
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
    rel="stylesheet" type="text/css">

    <style>

    .e-column:before {
        content: '\e815';
    }
    .e-header:before {
        content: '\ea9a';
    }
</style>

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <script id="customertemplate" type="text/x-template">
        <span class="e-icons e-header" ></span>
        Task Name
    </script>

    <div id="container">
        <button id="restore">Restore</button>
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Tool bar in ##Platform_Name## Gantt control

The Gantt control provides the toolbar support to handle Gantt actions. The [toolbar](#) property accepts the collection of built-in toolbar items and `ItemModel` objects for custom toolbar items.

Built-in toolbar items

Built-in toolbar items execute standard actions of the Gantt control, and these items can be added to toolbar by defining the [toolbar](#) as a collection of built-in items. It renders the button with icon and text.

The following table shows built-in toolbar items and its actions.

Built-in Toolbar Items	Actions
-----	-----
Add	Adds a new record.
Cancel	Cancels the edit state.
CollapseAll	Collapses all the rows.
Delete	Deletes the selected record.
Edit	Edits the selected record.
Indent	Indent the selected record to one level.
Outdent	Outdent the selected record to one level.
ExpandAll	Expands all the rows.
NextTimeSpan	Navigate the Gantt timeline to next time span.
PrevTimeSpan	Navigate the Gantt timeline to previous time span.

| Search | Searches the records by the given key. |

| Update | Updates the edited record. |

INDEX.JS

```
ej.gantt.Gantt.Inject(ej.gantt.Edit,ej.gantt.Toolbar);
var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    toolbar:
    ['Add','Cancel','CollapseAll','Delete','Edit','ExpandAll','NextTimeSpan','PreviousTimeSpan','Search','Update','Indent','Outdent'],
    editSettings: {
        allowEditing: true,
        allowAdding: true,
        allowDeleting: true
    }
});
ganttChart.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
```

```

    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

* The [toolbar](#) has options to define both built-in and custom toolbar items.

Custom toolbar items

Custom toolbar items can be added to the toolbar by defining the [toolbar](#) property as a collection of `ItemModels`.

Actions for this customized toolbar items are defined in the [toolbarClick](#) event.

By default, the custom toolbar items are at left position. You can change the position by using the `align` property. In the following sample, the `Quick Filter` toolbar item is positioned at right.

INDEX.JS

```

var clickHandler = function(args){
    if (args.item.id === 'toolbarfilter') {
        ganttChart.filterByColumn('TaskName', 'startswith', 'Identify');
    }
};
ej.gantt.Gantt.Inject(ej.gantt.Filter,ej.gantt.Toolbar);
var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    toolbarClick: clickHandler,
    toolbar: [{text: 'Quick Filter', tooltipText: 'Quick
Filter', id: 'toolbarfilter', align:'Right', prefixIcon: 'e-quickfilter'}],
    allowFiltering:true
});
ganttChart.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

* The [toolbar](#) has options to define both built-in and custom toolbar items.

* If a toolbar item does not match the built-in items, it will be treated as a custom toolbar item.

Built-in and custom items in toolbar

The Gantt control has an option to use both built-in and custom toolbar items at the same time.

In the following example, the **ExpandAll** and **CollapseAll** are built-in toolbar items and **Test** is the custom toolbar item.

INDEX.JS

```

var clickHandler = function(args){
    if (args.item.text === 'Test') {
        alert("Custom toolbar click...");
    }
};
ej.gantt.Gantt.Inject(ej.gantt.Filter,ej.gantt.Toolbar);
var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    toolbarClick: clickHandler,
    toolbar: ['ExpandAll', 'CollapseAll', { text: 'Test',
tooltipText: 'Test',id: 'Test' }]
});
ganttChart.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Enable/disable toolbar items

You can enable or disable the toolbar items by using the `enableItems` method.

INDEX.JS

```

var clickHandler = function(args){
  if (args.item.text === 'QuickFilter') {
    ganttChart.filterByColumn('TaskName', 'startswith', 'Identify');
  }
  if (args.item.text === 'ClearFilter') {
    ganttChart.clearFiltering();
  }
};
ej.gantt.Gantt.Inject(ej.gantt.Filter,ej.gantt.Toolbar);
var ganttChart = new ej.gantt.Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',

```

```

        progress: 'Progress',
        child: 'subtasks'
    },
    toolbarClick: clickHandler,
    toolbar: ['QuickFilter', 'ClearFilter'],
    allowFiltering: true
});
ganttChart.appendTo('#Gantt');
var enable = new ej.buttons.Button({}, '#enable');
var disable = new ej.buttons.Button({}, '#disable');
enable.element.onclick = function() {
    ganttChart.toolbarModule.enableItems([ganttChart.element.id +
    '_QuickFilter', ganttChart.element.id + '_ClearFilter'], true); // enable
    toolbar items.
};
disable.element.onclick = function() {
    ganttChart.toolbarModule.enableItems([ganttChart.element.id +
    '_QuickFilter', ganttChart.element.id + '_ClearFilter'], false); // disable
    toolbar items.
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
    rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="enable" class="e-flat">Enable</button>
        <button id="disable" class="e-flat">Disable</button>
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Add input elements to toolbar

In the Gantt toolbar, you can add EJ2 editor elements like numeric text box, drop-down list, and date picker controls. The following code snippets demonstrates how to add EJ2 editors to the Gantt toolbar.

INDEX.JS

```
ej.gantt.Gantt.Inject(ej.gantt.Toolbar);
var ganttChart = new ej.gantt.Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  toolbar: [{ type: 'Input', template: new ej.inputs.NumericTextBox({
    format: 'c2', value:1, width:150 }) }]
});
ganttChart.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
```



```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Context menu in ##Platform_Name## Gantt control

The Gantt control allows you to perform quick actions by using context menu. When right-clicking the context menu, the context menu options are shown. To enable this feature, set the [enableContextMenu](#) to true. The default context menu options are enabled using the [editSettings](#) property. The context menu options can be customized using the [contextMenuItems](#) property.

To use the context menu, inject the [ContextMenu](#) module into the Gantt control.

The default items are listed in the following table.

Items	Description
AutoFit	Auto-fits the current column.
AutoFitAll	Auto-fits all columns.
SortAscending	Sorts the current column in ascending order.
SortDescending	Sorts the current column in descending order.
TaskInformation	Edits the current task.
Add	Adds a new row to the Gantt.
Indent	Indent the selected record to one level.
Outdent	Outdent the selected record to one level.
DeleteTask	Deletes the current task.
Save	Saves the edited task.
Cancel	Cancels the edited task.
DeleteDependency	Deletes the current dependency task link.
Convert	Converts current task to milestone or vice-versa.

INDEX.JS

```
var ganttChart = new ej.gantt.Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    dependency: 'Predecessor',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  allowSorting: true,
  allowResizing: true,
  editSettings: {
    allowAdding: true,
```

```

        allowEditing: true,
        allowDeleting: true,
        allowTaskbarEditing: true
    },
    enableContextMenu: true
});
gantChart.appendTo('#ContextMenu');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="ContextMenu"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Custom context menu items

The custom context menu items can be added by defining the [contextMenuItems](#) as a collection of [contextMenuItemModel](#). Actions for the customized items can be defined in the [contextMenuClick](#) event and the visibility of customized items can be defined in the [contextMenuOpen](#) event.

To create custom context menu items for header area, define the target property as `.e-gridheader`.

The following sample shows context menu item for parent rows to expand or collapse child rows in the content area and a context menu item to hide columns in the header area.

INDEX.JS

```
var ganttChart = new ej.gantt.Gantt({
```

```

        dataSource: GanttData,
        height: '450px',
        taskFields: {
            id: 'TaskID',
            name: 'TaskName',
            startDate: 'StartDate',
            dependency: 'Predecessor',
            duration: 'Duration',
            progress: 'Progress',
            child: 'subtasks'
        },
        allowSorting: true,
        allowResizing: true,
        editSettings: {
            allowAdding: true,
            allowEditing: true,
            allowDeleting: true,
            allowTaskbarEditing: true
        },
        enableContextMenu: true,
        contextMenuItems: ['AutoFitAll', 'AutoFit', 'TaskInformation',
            'DeleteTask', 'Save', 'Cancel',
            'SortAscending', 'SortDescending', 'Add', 'DeleteDependency',
            'Convert',
            { text: 'Collapse the Row', target: '.e-content', id:
            'collapserow' },
            { text: 'Expand the Row', target: '.e-content', id: 'expandrow'
        },
        { text: 'Hide Column', target: '.e-gridheader', id:
        'hidecols' }
    ],
    contextMenuClick: function (args) {
        var record = args.rowData;
        if (args.item.id === 'collapserow') {
            ganttChart.collapseByID(Number(record.ganttProperties.taskId));
        }
        if (args.item.id === 'expandrow') {
            ganttChart.expandByID(Number(record.ganttProperties.taskId));
        }
        if (args.item.id === 'hidecols') {
            ganttChart.hideColumn(args.column.headerText);
        }
    },
    contextMenuOpen: function (args) {
        var record = args.rowData;
        if (args.type !== 'Header') {
            if (!record.hasChildRecords) {
                args.hideItems.push('Collapse the Row');
                args.hideItems.push('Expand the Row');
            } else {
                if (record.expanded) {
                    args.hideItems.push("Expand the Row");
                } else {
                    args.hideItems.push("Collapse the Row");
                }
            }
        }
    }
}

```

```

    }
  }
});
gantChart.appendTo('#CustomContextMenu');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="CustomContextMenu"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

You can show an specific item in context menu for header/content area in the Gantt control by defining the **target** property.

Excel Export

Excel export in ##Platform_Name## Gantt control

Gantt supports client-side exporting, which allows you to export its data to the Excel and CSV formats. Use the [excelExport](#) and [csvExport](#) methods for exporting. To enable Excel export in the Gantt, set the [allowExcelExport](#) to true.

To export data to Excel and CSV, inject the **ExcelExport** module in Gantt.

INDEX.JS

```

var clickHandler = function(args) {
  if (args.item.id === 'GanttExport_excelexport') {
```

```

        ganttChart.excelExport();
    } else if (args.item.id === 'GanttExport_csvexport') {
        ganttChart.csvExport();
    }
};
ej.gantt.Gantt.Inject(ej.gantt.ExcelExport, ej.gantt.Toolbar);
var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    allowExcelExport: true,
    toolbar: ['ExcelExport', 'CsvExport'],
    toolbarClick: clickHandler
});
ganttChart.appendTo('#GanttExport');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="GanttExport"></div>
    </div>

<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Custom data source in ##Platform_Name## Gantt control

The excel export provides an option to define datasource dynamically before exporting. To export data dynamically, define the `dataSource` in `exportProperties`.

INDEX.JS

```
var clickHandler = function(args){
    if (args.item.id === 'GanttExport_excelexport') {
        var excelExportProperties = {
            dataSource: [GanttData[1]]
        };
        ganttChart.excelExport(excelExportProperties);
    }
};
ej.gantt.Gantt.Inject(ej.gantt.ExcelExport,ej.gantt.Toolbar);
var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    allowExcelExport: true,
    toolbar: ['ExcelExport'],
    toolbarClick: clickHandler
});
ganttChart.appendTo('#GanttExport');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
```

```

        <div id="GanttExport"></div>
    </div>

<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Multiple gantt exporting in ##Platform_Name## Gantt control

In Gantt, the Excel export provides support to export multiple Gantt data in new sheet or same sheet.

Same sheet

The Excel export provides support to export multiple Gantt data in the same sheet. To export in same sheet, define `multipleExport.type` as `AppendToSheet` in `ExcelExportProperties`. You can also provide blank rows between exported multiple Gantt data. These blank rows count can be defined using `multipleExport.blankRows`.

INDEX.JS

```

ej.gantt.Gantt.Inject(ej.gantt.ExcelExport,ej.gantt.Toolbar);
var firstGantt = new ej.gantt.Gantt({
    dataSource: [GanttData[0]],
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    treeColumnIndex: 1,
    allowExcelExport: true,
    projectStartDate: new Date('03/31/2019'),
    projectEndDate: new Date('04/14/2019'),
    height:280,
    toolbar: ['ExcelExport']
});
firstGantt.appendTo('#GanttExport1');
var secondGantt = new ej.gantt.Gantt({
    dataSource: [GanttData[1]],
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    treeColumnIndex: 1,
    height:250,
    allowExcelExport: true

```

```

});
secondGantt.appendTo('#GanttExport2');
firstGantt.toolbarClick = function(args) {
    if (args.item.id === 'GanttExport1_excelexport') {
        var appendExcelExportProperties = {
            multipleExport: { type: 'AppendToSheet', blankRows: 2 }
        };
        var firstGanttExport=
firstGantt.excelExport(appendExcelExportProperties, true);
        firstGanttExport.then(function(fData) {
            secondGantt.excelExport(appendExcelExportProperties, false,
fData);
        });
    }
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="GanttExport1"></div>
        <div id="GanttExport2" style="margin-top:10px"></div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

By default, `multipleExport.blankRows` value is 5.

New sheet

The Excel exporting provides support to export multiple Gantt in new sheet. To export in new sheet, define `multipleExport.type` as `NewSheet` in `ExcelExportProperties`.

INDEX.JS

```

ej.gantt.Gantt.Inject(ej.gantt.ExcelExport,ej.gantt.Toolbar);
var firstGantt = new ej.gantt.Gantt({
  dataSource: [GanttData[0]],
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  treeColumnIndex: 1,
  allowExcelExport: true,
  projectStartDate: new Date('03/31/2019'),
  projectEndDate: new Date('04/14/2019'),
  height:280,
  toolbar: ['ExcelExport']
});
firstGantt.appendTo('#GanttExport1');
var secondGantt = new ej.gantt.Gantt({
  dataSource: [GanttData[1]],
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  treeColumnIndex: 1,
  height:250,
  allowExcelExport: true
});
secondGantt.appendTo('#GanttExport2');
firstGantt.toolbarClick = function(args) {
  if (args.item.id === 'GanttExport1_excelexport') {
    var appendExcelExportProperties = {
      multipleExport: { type: 'NewSheet' }
    };
    var firstGanttExport=
firstGantt.excelExport(appendExcelExportProperties, true);
    firstGanttExport.then(function(fData) {
      secondGantt.excelExport(appendExcelExportProperties, false,
fData);
    });
  }
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">

```

```

<meta name="description" content="Typescript Gantt Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="GanttExport1"></div>
    <div id="GanttExport2" style="margin-top:10px"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customize the Excel export

Gantt Excel export allows the users to customize the exported document based on requirement.

Export hidden columns

In Gantt, the Excel export provides an option to export hidden columns by defining `includeHiddenColumn` as `true`.

INDEX.JS

```

var clickHandler = function(args){
  if (args.item.id === 'GanttExport_excelexport') {
    var excelExportProperties = {
      includeHiddenColumn: true
    };
    ganttChart.excelExport(excelExportProperties);
  }
  else if (args.item.id === 'GanttExport_csvexport') {
    var excelExportProperties = {
      includeHiddenColumn: true
    };
    ganttChart.csvExport(excelExportProperties);
  }
};
ej.gantt.Gantt.Inject(ej.gantt.ExcelExport,ej.gantt.Toolbar);
var ganttChart = new ej.gantt.Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {

```

```

        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    treeColumnIndex: 1,
    allowExcelExport: true,
    columns: [
        { field: 'TaskID', headerText: 'Task ID', textAlign: 'Left', width:
'100' },
        { field: 'TaskName', headerText: 'Task Name', width: '150' },
        { field: 'StartDate', headerText: 'Start Date', width:
'150', visible: false },
        { field: 'Duration', headerText: 'Duration', width: '150' },
        { field: 'Progress', headerText: 'Progress', width: '150' },
    ],
    toolbar: ['ExcelExport', 'CsvExport'],
    toolbarClick: clickHandler
});
gantChart.appendTo('#GanttExport');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="GanttExport"></div>
  </div>

  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Show or hide columns on exported Excel

In Gantt, while exporting, you can show a hidden column or hide a visible column using the [toolbarClick](#) and [excelExportComplete](#) events.

In the `toolbarClick` event, using the `args.item.id` property, you can show or hide columns by setting the `column.visible` property to `true` or `false` respectively.

Similarly, in the `excelExportComplete` event, you can revert the columns visibility back to the previous state.

INDEX.JS

```
var clickHandler = function(args){
    let columns = ganttChart.treeGrid.grid.columns;
    if (args.item.id === 'GanttExport_excelexport') {
        columns[0].visible = true;
        columns[3].visible = false;
        ganttChart.excelExport();
    }
    else if (args.item.id === 'GanttExport_csvexport') {
        columns[0].visible = true;
        columns[3].visible = false;
        ganttChart.csvExport();
    }
};
var exportComplete = function(args){
    let columns = ganttChart.treeGrid.grid.columns;
    columns[0].visible = false;
    columns[3].visible = true;
};
ej.gantt.Gantt.Inject(ej.gantt.ExcelExport,ej.gantt.Toolbar);
var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    treeColumnIndex: 1,
    allowExcelExport: true,
    columns: [
        { field: 'TaskID', headerText: 'Task ID', textAlign: 'Left', width:
'100',visible: false },
        { field: 'TaskName', headerText: 'Task Name', width: '150' },
        { field: 'StartDate', headerText: 'Start Date', width: '150' },
        { field: 'Duration', headerText: 'Duration', width: '150' },
        { field: 'Progress', headerText: 'Progress', width: '150' },
    ],
    toolbar: ['ExcelExport', 'CsvExport'],
    toolbarClick: clickHandler,
    excelExportComplete: exportComplete
});
```

```
});
ganttChart.appendTo('#GanttExport');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="GanttExport"></div>
  </div>

  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Cell formatting during export

In Gantt, you can customize the TreeGrid cells in the exported document using the [excelQueryCellInfo](#) event. In this event, you can format the TreeGrid cells of exported Excel and CSV documents based on the required condition.

In the following sample, the background color has been customized for **TaskID** column in the exported Excel using the **args.style** and **backColor** properties.

INDEX.JS

```
var clickHandler = function(args){
  if (args.item.id === 'GanttExport_excelexport') {
    ganttChart.excelExport();
  }
};
ej.gantt.Gantt.Inject(ej.gantt.ExcelExport,ej.gantt.Toolbar);
var ganttChart = new ej.gantt.Gantt({
  dataSource: GanttData,
```

```

height: '450px',
taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
},
treeColumnIndex: 1,
allowExcelExport: true,
toolbar: ['ExcelExport'],
labelSettings:{
    taskLabel: '${Progress}%'
},
splitterSettings: {
    columnIndex: 3
},
columns: [
    { field: 'TaskID', headerText: 'Task ID', textAlign: 'Left', width:
'100',visible:false },
    { field: 'TaskName', headerText: 'Task Name', width: '150' },
    { field: 'Progress', headerText: 'Progress', width: '150' },
    { field: 'StartDate', headerText: 'Start Date', width: '150' },
    { field: 'Duration', headerText: 'Duration', width: '150' }
],
excelQueryCellInfo: function(args) {
    if(args.column.field == 'Progress'){
        if(args.value > 80) {
            args.style = { backgroundColor: '#A569BD' };
        }
        else if(args.value < 20) {
            args.style = { backgroundColor: '#F08080' };
        }
    }
},
queryTaskbarInfo: function(args) {
    if (args.data.Progress > 80) {
        args.progressBarBgColor = "#6C3483";
        args.taskbarBgColor = args.taskbarBorderColor = "#A569BD";
    } else if (args.data.Progress < 20) {
        args.progressBarBgColor = "#CD5C5C";
        args.taskbarBgColor = args.taskbarBorderColor = "#F08080";
    }
},
queryCellInfo: function(args) {
    if(args.column.field == 'Progress'){
        if(args.data.Progress > 80) {
            args.cell.style.backgroundColor = '#A569BD';
        }
        else if(args.data.Progress < 20) {
            args.cell.style.backgroundColor = '#F08080';
        }
    }
},
toolbarClick: clickHandler
});

```

```
ganttChart.appendTo('#GanttExport');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="GanttExport"></div>
  </div>

  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Theme

The Excel export also provides an option to include custom theme for exported Excel document.

To apply theme in exported Excel, define the **theme** in **ExcelExportProperties**.

INDEX.JS

```
ej.gantt.Gantt.Inject(ej.gantt.ExcelExport,ej.gantt.Toolbar);
var ganttChart = new ej.gantt.Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  treeColumnIndex: 1,
```

```

        allowExcelExport: true,
        toolbar: ['ExcelExport'],
    });
    ganttChart.appendTo('#GanttExport');
    ganttChart.toolbarClick = (args) => {
        if (args['item'].id === 'GanttExport_excelexport') {
            let excelExportProperties = {
                theme: {
                    header: { fontName: 'Segoe UI', fontColor: '#666666' },
                    record: { fontName: 'Segoe UI', fontColor: '#666666' },
                    caption: { fontName: 'Segoe UI', fontColor: '#666666' }
                }
            };
            ganttChart.excelExport(excelExportProperties);
        }
    }
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="GanttExport"></div>
  </div>

  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

By default, material theme is applied to the exported Excel document.

Add header and footer

The Excel export also allows users to include header and footer contents to the exported Excel document.

INDEX.JS

```
ej.gantt.Gantt.Inject(ej.gantt.ExcelExport,ej.gantt.Toolbar);
var ganttChart = new ej.gantt.Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  treeColumnIndex: 1,
  allowExcelExport: true,
  toolbar: ['ExcelExport'],
});
ganttChart.appendTo('#GanttExport');
ganttChart.toolbarClick = (args) => {
  if (args['item'].id === 'GanttExport_exceleport') {
    let excelExportProperties = {
      header: {
        headerRows: 7,
        rows: [
          { cells: [{ colSpan: 4, value: "Northwind Traders",
style: { fontColor: '#C67878', fontSize: 20, hAlign: 'Center', bold: true, }
}} ],
          { cells: [{ colSpan: 4, value: "2501 Aerial Center
Parkway", style: { fontColor: '#C67878', fontSize: 15, hAlign: 'Center',
bold: true, } } ] },
          { cells: [{ colSpan: 4, value: "Suite 200 Morrisville,
NC 27560 USA", style: { fontColor: '#C67878', fontSize: 15, hAlign:
'Center', bold: true, } } ] },
          { cells: [{ colSpan: 4, value: "Tel +1 888.936.8638 Fax
+1 919.573.0306", style: { fontColor: '#C67878', fontSize: 15, hAlign:
'Center', bold: true, } } ] },
          { cells: [{ colSpan: 4, hyperlink: { target:
'https://www.northwind.com/', displayText: 'www.northwind.com' }, style: {
hAlign: 'Center' } } ] },
          { cells: [{ colSpan: 4, hyperlink: { target:
'mailto:support@northwind.com' }, style: { hAlign: 'Center' } } ] },
        ]
      },
      footer: {
        footerRows: 4,
        rows: [
          { cells: [{ colSpan: 4, value: "Thank you for your
business!", style: { hAlign: 'Center', bold: true } } ] },
          { cells: [{ colSpan: 4, value: "!Visit Again!", style: {
hAlign: 'Center', bold: true } } ] }
        ]
      }
    },
  }
}
```

```

    };
    ganttChart.excelExport(excelExportProperties);
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="GanttExport"></div>
  </div>

  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

File name for exported document

You can set the required file name for the exported document by defining the `fileName` property in `ExcelExportProperties`.

INDEX.JS

```

ej.gantt.Gantt.Inject(ej.gantt.ExcelExport,ej.gantt.Toolbar);
var ganttChart = new ej.gantt.Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',

```

```

        child: 'subtasks'
    },
    treeColumnIndex: 1,
    allowExcelExport: true,
    toolbar: ['ExcelExport', 'CsvExport'],
});
gantChart.appendTo('#GanttExport');
gantChart.toolbarClick = (args) => {
    if (args['item'].id === 'GanttExport_excelexport') {
        let excelExportProperties = {
            fileName: "Gantt.xlsx"
        };
        gantChart.excelExport(excelExportProperties);
    }
    else if (args['item'].id === 'GanttExport_csvexport') {
        let excelExportProperties = {
            fileName: "Gantt.csv"
        };
        gantChart.csvExport(excelExportProperties);
    }
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="GanttExport"></div>
    </div>

    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

Pdf Export

Pdf export in ##Platform_Name## Gantt control

PDF Export

PDF export allows exporting Gantt data to PDF document. You need to use the [pdfExport](#) method for exporting. To enable PDF export in the Gantt, set the [allowPdfExport](#) to true.

To export data to PDF document, inject the PdfExport module in Gantt.

INDEX.JS

```
var clickHandler = function(args){
    if (args.item.id === 'GanttExport_pdfexport') {
        ganttChart.pdfExport();
    }
};
var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    allowPdfExport: true,
    toolbar: ['PdfExport'],
    toolbarClick: clickHandler
});
ganttChart.appendTo('#GanttExport');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="GanttExport"></div>
```

```

        </div>

<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Indicators in PDF exporting

The PDF export functionality allows users to export Gantt charts enriched with dynamic indicators and accompanying images.

INDEX.JS

```

var clickHandler = function(args){
    if (args.item.id === 'GanttExport_pdfexport') {
        ganttChart.pdfExport();
    }
};
var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks',
        indicators: 'Indicators'
    },
    allowPdfExport: true,
    toolbar: ['PdfExport'],
    toolbarClick: clickHandler
});
ganttChart.appendTo('#GanttExport');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>

```

```

<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="GanttExport"></div>
    </div>

<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Exporting Gantt data as a blob object

In Gantt, you can export the Gantt chart data as a blob object, which allows you to preview or modify the data before exporting it.

To export the Gantt chart data as a blob object, follow these steps:

step 1: pdfExport fourth argument set as **true**.

step 2: Then , pdfExpComplete return as blob object.

INDEX.JS

```

/**
 * Exporting Blob data
 */
let excelExpComplete = (args) => {
    //This event will be triggered when excel exporting.
    args.promise.then((e) => {
        //In this `then` function, we can get blob data through the arguments
        //after promise resolved.
        exportBlob(e.blobData);
    });
};

let pdfExpComplete = (args) => {
    //This event will be triggered when pdf exporting.
    args.promise.then((e) => {
        //In this `then` function, we can get blob data through the arguments
        //after promise resolved.
        exportBlob(e.blobData);
    });
};

let exportBlob = (blob) => {
    let a = document.createElement('a');
    document.body.appendChild(a);
    a.style.display = 'none';
    let url = window.URL.createObjectURL(blob);
    a.href = url;
    a.download = 'Export';

```

```

a.click();
window.URL.revokeObjectURL(url);
document.body.removeChild(a);
}
const clickHandler = (args) => {
if (args.item.id === 'GanttExport_pdfexport') {
    gantt.pdfExport(null, null, null, true);
}
if (args.item.id === 'GanttExport_excelexport') {
    gantt.excelExport(null, null, null, true);
}
};

var gantt = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    allowPdfExport: true,
    allowExcelExport: true,
    excelExportComplete: excelExpComplete,
    pdfExportComplete: pdfExpComplete,
    toolbar: ['PdfExport', 'ExcelExport'],
    toolbarClick: clickHandler
});
gantt.appendTo('#GanttExport');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">
  <script src="http://cdn.syncfusion.com/ej2/20.4.48/dist/ej2.min.js"
type="text/javascript"></script>

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```

```

        <div id="container">
            <div id="GanttExport"></div>
        </div>

</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Single page exporting in gantt

In Gantt, we have provided support to export the Gantt component where each rows are auto-fit to the PDF document page width by setting [isFitToWidth](#) as true in `fitToWidthSettings` of `PdfExportProperties`.

Also, we can customize the chart width and grid width in exported file using [chartWidth](#) and [gridWidth](#) by defining it as percentage in string.

INDEX.JS

```

var clickHandler = function(args){
    if (args.item.id === 'GanttExport_pdfexport') {
        var exportProperties = {
            fitToWidthSettings: {
                isFitToWidth: true,
            }
        };
        ganttChart.pdfExport(exportProperties);
    }
};

var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    dateFormat: 'MMM dd, y',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        endDate: 'EndDate',
        duration: 'Duration',
        progress: 'Progress',
        dependency: 'Predecessor',
        child: 'subtasks',
    },
    columns: [
        { field: 'TaskID', width: 80 },
        { field: 'TaskName', width: 250 },
        { field: 'StartDate' },
        { field: 'EndDate' },
        { field: 'Duration' },
        { field: 'Predecessor' },
        { field: 'resources' },
        { field: 'Progress' }
    ]
});

```



```

    ],
    splitterSettings: {
        columnIndex: 2
    },
    allowPdfExport: true,
    toolbar: ['PdfExport'],
    toolbarClick: clickHandler,
    allowSelection: true,
    gridLines: 'Both',
    height: '450px',
    treeColumnIndex: 1,
    resourceFields: {
        id: 'resourceId',
        name: 'resourceName'
    },
    highlightWeekends: true,
    timelineSettings: {
        topTier: {
            unit: 'Week',
            format: 'MMM dd, y',
        },
        bottomTier: {
            unit: 'Day',
        },
    },
    labelSettings: {
        leftLabel: 'TaskName',
        rightLabel: 'resources'
    },
    projectStartDate: new Date('03/25/2019'),
    projectEndDate: new Date('07/28/2019')
});
gantChart.appendTo('#GanttExport');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">

```

```

        <div id="GanttExport"></div>
    </div>

    <script>
    var ele = document.getElementById('container');
    if(ele) {
        ele.style.visibility = "visible";
    }
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

Exporting with column template

The PDF export functionality allows to export Grid columns that include images, hyperlinks, and custom text to an PDF document using [pdfQueryCellInfo](#) event.

In the following sample, the hyperlinks and images are exported to PDF using [hyperlink](#) and [image](#) properties in the [pdfQueryCellInfo](#) event.

Note: PDF Export supports base64 string to export the images.

INDEX.JS

```

var clickHandler = function(args){
    if (args.item.id === 'GanttExport_pdfexport') {
        var exportProperties = {
            enableFooter: false
        };
        ganttChart.pdfExport(exportProperties);
    }
};
function pdfQueryCellInfo(args) {
    if (args.column.headerText === 'Resources') {
        {
            args.image = { height: 40, width: 40, base64:
args.data.taskData.resourcesImage };
        }
    }
    if (args.column.headerText === 'Email ID') {
        args.hyperLink = {
            target: 'mailto:' + args.data.taskData.EmailId,
            displayText: args.data.taskData.EmailId
        };
    }
}
var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        endDate: 'EndDate',
        duration: 'Duration',
        progress: 'Progress',
        resourceInfo: 'resources',

```

```

        dependency: 'Predecessor',
        child: 'subtasks'
    },
    allowPdfExport: true,
    columns: [
        { field: 'TaskID', headerText: 'Task ID', textAlign: 'Left' },
        { field: 'TaskName', headerText: 'Task Name', width: '250' },
        { field: 'resources', headerText: 'Resources', width: '250',
template: '#columnTemplate' },
        {field: 'EmailId', headerText: 'Email ID', template: '#template2',
width: 180 },
    ],
    pdfQueryCellInfo: pdfQueryCellInfo,
    toolbar: ['PdfExport'],
    toolbarClick: clickHandler,
    resources: editingResources,
    resourceFields: {
        id: 'resourceId',
        name: 'resourceName'
    },
    projectStartDate: new Date('03/24/2019'),
    projectEndDate: new Date('07/06/2019')
});
gantChart.appendTo('#GanttExport');
```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css" rel="stylesheet"
type="text/css">
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <script type="text/x-jsrender" id="columnTemplate">
        ${if(ganttProperties.resourceNames) }
        <div class="image">
            
        </div>
        ${/if}
    </script>
    <script id="template2" type="text/x-template">
        ${if(taskData.EmailId) }
        <div class="link">
```

```

        <a href="mailto:${taskData.EmailId}">${taskData.EmailId}</a></div>
    </div>
    ${/if}
</script>
<div id="container">
    <div id="GanttExport"></div>
</div>
<script>
    var ele = document.getElementById('container');
    if (ele) {
        ele.style.visibility = "visible";
    }
</script>
<script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Multiple gantt exporting in ##Platform_Name## Gantt control

PDF export provides an option for exporting multiple Gantt to same file. In this exported document, each Gantt will be exported to a new page of the document in same file.

INDEX.JS

```

var firstGantt = new ej.gantt.Gantt({
    dataSource: [GanttData[0]],
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    treeColumnIndex: 1,
    allowExcelExport: true,
    projectStartDate: new Date('03/31/2019'),
    projectEndDate: new Date('04/14/2019'),
    height: 280,
    toolbar: ['ExcelExport']
});
firstGantt.appendTo('#GanttExport1');
var secondGantt = new ej.gantt.Gantt({
    dataSource: [GanttData[1]],
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    treeColumnIndex: 1,
    height: 250,
    allowExcelExport: true
});
secondGantt.appendTo('#GanttExport2');

```

```

firstGantt.toolbarClick = function(args) {
    if (args.item.id === 'GanttExport1_pdfexport') {
        var appendExcelExportProperties = {
            multipleExport: { type: 'AppendToSheet', blankRows: 2 }
        };
        var firstGanttExport= firstGantt.pdfExport({}, true);
        firstGanttPdfExport.then(function(pdfData) {
            secondGantt.pdfExport({}, false, pdfData);
        });
    }
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="GanttExport1"></div>
        <div id="GanttExport2" style="margin-top:10px"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

To customize PDF export

PDF export provides an option to customize the mapping of Gantt to exported PDF document.

File name for exported document

You can assign a file name for the exported document by defining the `fileName` property in `pdfExportProperties`.

INDEX.JS

```

var clickHandler = function(args){
    if (args.item.id === 'GanttExport_pdfexport') {
        var exportProperties = {
            fileName:"new.pdf"
        };
        ganttChart.pdfExport(exportProperties);
    }
};
var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    allowPdfExport: true,
    toolbar: ['PdfExport'],
    toolbarClick: clickHandler
});
ganttChart.appendTo('#GanttExport');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="GanttExport"></div>
    </div>

<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>

```

```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

How to change page orientation

Page orientation can be changed to **Portrait** (Default Landscape) for the exported document using the property `pdfExportProperties.pageOrientation`.

INDEX.JS

```
var clickHandler = function(args){
    if (args.item.id === 'GanttExport_pdfexport') {
        var exportProperties = {
            pageOrientation: 'Portrait'
        };
        ganttChart.pdfExport(exportProperties);
    }
};
var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    allowPdfExport: true,
    toolbar: ['PdfExport'],
    toolbarClick: clickHandler
});
ganttChart.appendTo('#GanttExport');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
```

```
<div id="container">
  <div id="GanttExport"></div>
</div>

<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

[How to change page size](#)

Page size can be customized for the exported document using the `pdfExportProperties.pageSize`.

The supported page sizes are:

- Letter
- Note
- Legal
- A0
- A1
- A2
- A3
- A5
- A6
- A7
- A8
- A9
- B0
- B1
- B2
- B3
- B4
- B5
- Archa
- Archb
- Archc
- Archd
- Arche
- Flsa
- HalfLetter
- Letter11x17
- Ledger

INDEX.JS

```
var clickHandler = function(args){
  if (args.item.id === 'GanttExport_pdfexport') {
```



```

        var exportProperties = {
            pageSize: 'A0'
        };
        ganttChart.pdfExport(exportProperties);
    }
};
var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    allowPdfExport: true,
    toolbar: ['PdfExport'],
    toolbarClick: clickHandler
});
ganttChart.appendTo('#GanttExport');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="GanttExport"></div>
    </div>

    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

Export current view data

PDF export provides an option to export the current view data into PDF. To export current view data alone, define the `exportType` to `CurrentViewData`.

INDEX.JS

```
var clickHandler = function(args){
    if (args.item.id === 'GanttExport_pdfexport') {
        var exportProperties = {
            exportType: 'CurrentViewData'
        };
        ganttChart.pdfExport(exportProperties);
    }
};
var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    allowFiltering: true,
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    allowPdfExport: true,
    toolbar: ['PdfExport'],
    toolbarClick: clickHandler
});
ganttChart.appendTo('#GanttExport');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
```

```

        <div id="GanttExport"></div>
    </div>

<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Enable footer

By default, we render the default footer for a PDF file, this can be enabled or disabled by using the `enableFooter` property.

INDEX.JS

```

var clickHandler = function(args){
    if (args.item.id === 'GanttExport_pdfexport') {
        var exportProperties = {
            enableFooter: false
        };
        ganttChart.pdfExport(exportProperties);
    }
};
var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    allowPdfExport: true,
    toolbar: ['PdfExport'],
    toolbarClick: clickHandler
});
ganttChart.appendTo('#GanttExport');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="GanttExport"></div>
    </div>

<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Export hidden columns

PDF export provides an option to export hidden columns of Gantt by defining the `includeHiddenColumn` to `true`.

INDEX.JS

```

var clickHandler = function(args){
    if (args.item.id === 'GanttExport_pdfexport') {
        var exportProperties = {
            includeHiddenColumn: true
        };
        ganttChart.pdfExport(exportProperties);
    }
};

var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    columns: [
        { field: 'TaskID'},
        { field: 'TaskName', visible: false},
        { field: 'StartDate'},
        { field: 'Duration'},
        { field: 'Progress'}
    ],
    allowPdfExport: true,
    toolbar: ['PdfExport'],

```

```

        toolbarClick: clickHandler
    });
    ganttChart.appendTo('#GanttExport');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="GanttExport"></div>
  </div>

  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Export predecessor lines

By using `showPredecessorLines`, you can hide or show predecessor lines in the exported PDF document.

INDEX.JS

```

var clickHandler = function(args){
  if (args.item.id === 'GanttExport_pdfexport') {
    var exportProperties = {
      showPredecessorLines: true
    };
    ganttChart.pdfExport(exportProperties);
  }
};
var ganttChart = new ej.gantt.Gantt({
  dataSource: GanttData,
  height: '450px',

```

```

        taskFields: {
            id: 'TaskID',
            name: 'TaskName',
            startDate: 'StartDate',
            duration: 'Duration',
            progress: 'Progress',
            child: 'subtasks'
        },
        allowPdfExport: true,
        toolbar: ['PdfExport'],
        toolbarClick: clickHandler
    });
    ganttChart.appendTo('#GanttExport');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="GanttExport"></div>
    </div>

    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

Show or hide columns on exported PDF

You can show a hidden column or hide a visible column while exporting the Gantt using the [toolbarClick](#) and [beforePdfExport](#) events.

You can show or hide columns by setting the `column.visible` property to `true` or `false` respectively.

In the following example, there is a hidden column **Duration** in the Gantt. While exporting, we have changed **Duration** to visible column and **StartDate** to hidden column.

INDEX.JS

```
var clickHandler = function(args){
    if (args.item.id === 'GanttExport_pdfexport') {
        var obj = document.getElementById('GanttExport').ej2_instances[0];
        obj.treeGrid.columns[2].visible = false;
        ganttChart.pdfExport();
    }
};
var beforePdfExport = function(args){
    var obj = document.getElementById('GanttExport').ej2_instances[0];
    obj.treeGrid.columns[3].visible = true;
};
var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    columns: [
        { field: 'TaskID' },
        { field: 'TaskName' },
        { field: 'StartDate' },
        { field: 'Duration', visible: false },
        { field: 'Progress' }
    ],
    allowPdfExport: true,
    toolbar: ['PdfExport'],
    toolbarClick: clickHandler,
    beforePdfExport: beforePdfExport
});
ganttChart.appendTo('#GanttExport');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
```

```

<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="GanttExport"></div>
    </div>

<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Conditional cell formatting

TreeGrid cells in the exported PDF can be customized or formatted using the [pdfQueryCellInfo](#) event. In this event, you can format the treegrid cells of exported PDF document based on the column cell value.

In the following sample, the background color is set for **Progress** column in the exported document by using the `args.style` and `backgroundColor` properties.

INDEX.JS

```

var clickHandler = function(args){
    if (args.item.id === 'GanttExport_pdfexport') {
        ganttChart.pdfExport();
    }
};
var pdfQueryCellInfo = function(args){
    if(args.column.field === 'Progress'){
        if(args.value < 50) {
            args.style = {backgroundColor: '#F08080'};
        } else {
            args.style = {backgroundColor: '#A569BD'};
        }
    }
};
var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    columns: [
        { field: 'TaskID'},
        { field: 'TaskName'},

```



```

        { field: 'StartDate'},
        { field: 'Duration', visible: false},
        { field: 'Progress'}
    ],
    allowPdfExport: true,
    toolbar: ['PdfExport'],
    toolbarClick: clickHandler,
    pdfQueryCellInfo: pdfQueryCellInfo
});
gantChart.appendTo('#GanttExport');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="GanttExport"></div>
  </div>

  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Timeline cell formatting

Timeline cells in the exported PDF document can be customized or formatted using the [pdfQueryTimelineCellInfo](#) event.

In the following sample, the header background color is set for timeline cells in the exported document by using the `args.headerBackgroundColor` property.

INDEX.JS

```
var clickHandler = function(args) {
```

```

    if (args.item.id === 'GanttExport_pdfexport') {
        ganttChart.pdfExport();
    }
};
var pdfQueryTimelineCellInfo = function(args){
    if(args.column.field === 'Progress'){
        args.timelineCell.backgroundColor = new ej.pdfexport.PdfColor(240,
248, 255);
    }
};
var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    columns: [
        { field: 'TaskID'},
        { field: 'TaskName'},
        { field: 'StartDate'},
        { field: 'Duration', visible: false},
        { field: 'Progress'}
    ],
    allowPdfExport: true,
    toolbar: ['PdfExport'],
    toolbarClick: clickHandler,
    pdfQueryTimelineCellInfo: pdfQueryTimelineCellInfo
});
ganttChart.appendTo('#GanttExport');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```

```

        <div id="container">
            <div id="GanttExport"></div>
        </div>

<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Taskbar formatting

Taskbars in the exported PDF document can be customized or formatted using the [pdfQueryTaskbarInfo](#) event.

In the following sample, the taskbar background color is customized in the chart side of the exported document by using the `args.taskbar` property.

INDEX.JS

```

var clickHandler = function(args){
    if (args.item.id === 'GanttExport_pdfexport') {
        ganttChart.pdfExport();
    }
};

var pdfQueryTaskbarInfo = function(args){
    if(args.data.Progress < 50 && !args.data.hasChildRecords) {
        args.taskbar.progressColor = new ej.pdfexport.PdfColor(205, 92, 92);
        args.taskbar.taskColor = args.taskbar.taskBorderColor = new
ej.pdfexport.PdfColor(240, 128, 128);
    }
};

var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    columns: [
        { field: 'TaskID' },
        { field: 'TaskName' },
        { field: 'StartDate' },
        { field: 'Duration', visible: false },
        { field: 'Progress' }
    ],
    allowPdfExport: true,
    toolbar: ['PdfExport'],
    toolbarClick: clickHandler,
    pdfQueryTaskbarInfo: pdfQueryTaskbarInfo

```

```
});
ganttChart.appendTo('#GanttExport');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="GanttExport"></div>
  </div>

  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Theme

PDF export provides an option to include theme for the exported PDF document. To apply theme in exported PDF, define the **theme** in **pdfExportProperties**. The available themes are:

- Material
- Fabric
- Bootstrap
- Bootstrap 4

INDEX.JS

```
var clickHandler = function(args){
  if (args.item.id === 'GanttExport_pdfexport') {
    var exportProperties = {
      theme:"Fabric"
    };
  }
};
```

```

        ganttChart.pdfExport (exportProperties);
    }
};
var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    allowPdfExport: true,
    toolbar: ['PdfExport'],
    toolbarClick: clickHandler
});
ganttChart.appendTo('#GanttExport');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="GanttExport"></div>
    </div>

    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

Customized Theme

PDF export provides an option to customize the Gantt style for the exported PDF document. To customize Gantt style in exported PDF, define the `ganttStyle` in `pdfExportProperties`.

INDEX.JS

```

var clickHandler = function(args){
    if (args.item.id === 'GanttExport_pdfexport') {
        var exportProperties = {
            ganttStyle: {
                fontFamily: 1,
                columnHeader: {
                    backgroundColor: new ej.pdfexport.PdfColor(179, 219,
255)
                },
                taskbar: {
                    taskColor: new ej.pdfexport.PdfColor(240, 128, 128),
                    taskBorderColor: new ej.pdfexport.PdfColor(240, 128,
128),
                    progressColor: new ej.pdfexport.PdfColor(205, 92, 92),
                },
                connectorLineColor: new ej.pdfexport.PdfColor(128, 0, 0),
                footer: {
                    backgroundColor: new ej.pdfexport.PdfColor(205, 92, 92)
                },
                timeline: {
                    backgroundColor: new ej.pdfexport.PdfColor(179, 219,
255),
                    fontStyle: 1
                },
                label: {
                    fontColor: new ej.pdfexport.PdfColor(128, 0, 0),
                },
                cell: {
                    backgroundColor: new ej.pdfexport.PdfColor(240, 248,
255),
                    fontColor: new ej.pdfexport.PdfColor(0, 0, 0),
                    borderColor: new ej.pdfexport.PdfColor(179, 219, 255),
                },
            }
        };
        ganttChart.pdfExport(exportProperties);
    }
};
var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    allowPdfExport: true,
    toolbar: ['PdfExport'],

```

```

        toolbarClick: clickHandler
    });
    ganttChart.appendTo('#GanttExport');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="GanttExport"></div>
  </div>

  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Customizing header and footer of PDF export in ##Platform_Name## Gantt control

PDF export provides an option to specify and customize text, page number, line and image in header and footer of exported PDF document by using [pdfExportProperties](#).

Write a text in header and footer

This functionality helps to customize the text that appears in the header or footer sections of a PDF document. Text can be added to [header](#) or [footer](#) of exported PDF document by using [pdfExportProperties](#).

- **type** property in the content array indicates the content type, such as 'Text'.
- **Value** property determines the text.
- **Position** property determines the horizontal and vertical positions of the text element.
- **style** property define the visual styling properties for the text element

`ts

```
let exportProperties: PdfExportProperties = {
  header: {
    fromTop: 0,
    height: 130,
    contents: [
      {
        type: 'Text',
        value: 'INVOICE',
        position: { x: 380, y: 0 },
        style: { textBrushColor: '#C25050', fontSize: 25 },
      },
    ]
  }
},
```

Draw a line in header and footer

This functionality helps to customize the line that appears in the header or footer sections of the PDF document. A line can be added to [header](#) or [footer](#) of the exported PDF document by using [pdfExportProperties](#).

- **type** determines content type, such as 'Line'.
- **style** is used to set properties like the color (penColor), size (penSize), and style (dashStyle) of the line.
- **points** specifies the coordinates for the start and end points of the line.

Supported line styles:

- dash
- dot
- dashdot
- dashdotdot
- solid

`ts

```
let exportProperties: PdfExportProperties = {
  header: {
    fromTop: 0,
    height: 130,
    contents: [
```



```
{
  type: 'Line',
  style: { penColor: '#000080', penSize: 2, dashStyle: 'Solid' },
  points: { x1: 0, y1: 4, x2: 685, y2: 4 }
}
]
}
}
```

Add page number in header and footer

This feature allows to customize the page number that appears in the header or footer sections of the PDF document. Page numbers can be added in [header](#) or [footer](#) of the exported PDF document by using [pdfExportProperties](#).

- **type** indicates that the content is a page number.
- **pageNumberType** specifies the type of numbering to be used.
- **format** is an optional attribute that allows you to customize the text format of the page number.
- **position** defines the coordinates (x, y) where the page number will be located.
- **style** sets the styling properties of the page number text, such as color (textBrushColor), font size (fontSize), and horizontal alignment (hAlign).

Supported page number types:

- LowerLatin - a, b, c,
- UpperLatin - A, B, C,
- LowerRoman - i, ii, iii,
- UpperRoman - I, II, III,
- Number - 1,2,3.

```
`ts
```

```
let exportProperties: PdfExportProperties = {
  header: {
    fromTop: 0,
    height: 130,
    contents: [
      {
        type: 'PageNumber',
        pageNumberType: 'Arabic',
```

```

format: 'Page {$current} of {$total}', //optional
position: { x: 0, y: 25 },
style: { textBrushColor: '#ffff80', fontSize: 15, hAlign: 'Center' }
}
]
}
}
,

```

Insert an image in header and footer

This feature allows to customize the image that appears in the header or footer sections of the PDF document. Image (Base64 string) can be added in the exported document in [header](#) or [footer](#) of the exported PDF document by using [pdfExportProperties](#).

- **type** indicates that the content is an image.
- **src** specifies the source of the image, which should be Base64 string.
- **Position** determines the horizontal and vertical positions of the image will be located.
- **size** sets the dimensions of the image.

Note: PDF Export supports base64 string to export the images.

```

`ts
// Replace it with a valid Base64-encoded image.
let image: string = "/9j/4AAQSkZJRgABAQEAAeAB4AAD..."
let exportProperties: PdfExportProperties = {
  header: {
    fromTop: 0,
    height: 130,
    contents: [
      {
        type: 'Image',
        src: image,
        position: { x: 40, y: 10 },
        size: { height: 100, width: 250 },
      }
    ]
  }
}

```

INDEX.JS

3359

uTm4sBD4dn6jxv5YPwCls3xt+ZL6gfYPcpauDvyuGvlVOyuTcbUcp1DklkyN6XU1TL1LprSMv3C9
FafNUR1IUfNyNzfFWW2zCD1Er2hRWRN4sknF9d3chmsVJpnN324uDcNp5CYN5ZJUW1UkmR/wAhr2e
r3QVGzL5SYikPYbCAx5sOsNIVqlao0tOwGzEGGC1wtG6aMHOV0+aBut7uRH9A/wd4DF/Mirnpnf2
4nyrI2kU16Ifub3pzQN1vdyI/oH/8A/Mirnpnf24nyptIpr0Q/c3vXr2WvWsRbKaHJrPzN1+LJpT
2QuFdbLISZEZ1Uki98Qk6IrhRFOTF6yMQufYTYWubgFluFwAyhYFJVapKiYN8TTAG22b5pwnUTmW
XizqBQEQUEUe4Q2wPeHvYmXq6xmUfwuFyh1rpf3h1KjFvpae9IbXONQgXIFygIrS8WF2Os23yxvBM
igVo4aOsO1Scn4vnVYE669zPw6I4VQvjN43UOpRhxrpujuiAisCxTvVF5veyN7yKfW3FB9rsWdJed
zKQL1Nkm0nh6/sIfEtbvL85y/9QvpKrfkiW5I7ViorymkBFKODhskH4tiP7mxsHwY+X/6b/8AJip
dffJHtt6nLaUfQ60sgIgIo9whtge8PexMvV1jMo/hcL1DrXSLvDqVGLfS096Q2ucahAuQL1ARW14
sLsdZtv1ljeCZFArRw0ckdqk5PxfOqWJ117mfh0RwqhfGbxuodSjDjXTHdEBFYFineqLze9lP3kU+
tuKD7XYS6S87mUgXqbJNpPD1/YQ+Ja3eX5zl/6hfSVW/JETyR2rFRX1NICKUcHDZIPxbEf3NjYPg
x8v8A9N/+TFS6++SPbb1OW0o+hlpZARARR7hDbA94e9iZerrGZR/C4XKHwukXeHUqMW+lp70htc4
1CBcgXKAitLxYXY6zbfLG8EyKBWjho5I7VJyfi+dVgTrr3M/DojhVC+M3jdQ61GHGumO6ICKWlFO
9UXm97KfvIp9bcUH2uxZ0153MpAvU2SbSeHr+wh8Slu8vznL/ANQvpKrfkiW5I7ViorymkBFKODh
skH4tiP7mxsHwY+X/AOm//JipdffJHtt6nLaUfQ60sgIgIvJtZCWbj7LzeCtjrXkE/BPNzPXTmps
61NBk7lqqWSnJrU6lQh6Qi9sRph763BrXBsIw4lrCV2GLXpmK62nj9HHCEvqnPX6D3LGuJbQnsYY
tfcut9P044cX1Tnr9H0S5l9Cexhi19y630+jjgvqnPX6PolzL6FONycjuRkFkYiAuI5Acrpxri3+
Q0YUQwUSaU5dVEpVFZORUq7gip581EiAzdt1Zlx2L2hhgG4xKE4q7HFvnFPKiyuv1c3FG7lT5BK
6nlV9u01rUSgmqbsF13Zq+i8SyX0L5exhi19y630+jjhzfVOev0fRLmX0J7GGLX3Lrft6OOC+qc9
fo+iXMv0Us3DWXwabOKnXM9cquVEFD8luQcwTENmy9R1Si1ZOLymiucR8/Fnolzft11stFmtesIQ
xbsdi69rJ7grQtpJjD2ytbYiGnjbxlHMxk5aafQ7QjotBuEaToZzjIVWYqFRLJxTNxpIPc/CXWE2
qegVppKThtgQpgta3ABgwe5eTyyYGP773eenmeMHj+WLEfy8ftK9duVLfxR93cnLJgY/vvd56eZ4
wPy0oj+Xj9pTb1S38Ufd3LKLuZ1g6xt09bXX2mslHT04datRlc0biHzZI05Z5CVmeSR5NTptkMuS
qdIUDEvqWlBCdZzBYRgOTDqWLOVhn6U7hBMxi9tttmdGpREootARARYBhA7BN4e9aaequDLo/hcL
lDrXSJvDqVFraEamnoE+5LaG2ScKhAuWQjts8g4XKZCO1LyAitKxYBEWDxOiIiIuWam0fwGBQa0c
NbyR1lSun4vnVYU8Qjk9NTyS64RO19KoxXh3DdQ6lGnGV0shHal5B2RMhHal5ARB/AOKdIijryyI
iL2uV6P8AeIFPrbig+12LOkv05lqlhcISeFDeeZpKpz9e19C0LBRB/wCvg8ntKxY/jXa+xRLkI7U
vIJBeaZCO1LyAi2rxZ6UlhOkZJiv1ZmO19LDiv1n4B7Q6ismT8bzdytkGvFKICICLAMIHYJvD3rT
T1VwZdH8Lhcoda6RN4dSoub6WnvSG2DjUIFYBcoCK0nFgdjxOt88ZwDAoNaOGt5I6ypKT8XzqsOe
dfZr4wieFUL3D3jdQ6lGnGukO6ICLf/FPdXX1fw5X9sQKfWzFC9rsWdJedzLvrC47KC8/x+vgWhP
0R5Pg8ntKxY/jXa+xRKJFfaAi2rxZ/ZOFvZmPcw4r9ZuAe0OorJk/G83crYxrxSiAiAiWDCB2CbW
96009VcGXR/C4XKHwukTeHUqLm+lp70htg41CBcgXKAitJxYHY8TrfPGcAwKDWjhreS0sqSk/F86
rDnnX2a+MInhVC9w943UOpRpxrpDuiAi3/wAU9ldeV/Dlf2xAp9bMUL2uxZ0153MtWsLjsoLz/H6
+BaE/RHk+Dye0rFj+Ndr7FEokV5oCLavFn9k4W9mY8LDiv1m4B7Q6ismT8bzdytjGvFKICICLybW
2Zl1tLLTiyE3U8mBncC/L4k2VZLhNOoNCsk6HQ6KOhj0hRXQYjYjcYNvQuHC6FhWrScWDg5pSSSm
VsqEVOuiOKE/tonszeH15w9K/edhYONDk2PpRHFbtonszej6pecPSnOwsHTulbH0ojig20T2ZvR
9UvOHpU63H3E2LuAsfE2Js09MnJfFRrketUe+TrmqRqhJ0USSzUQWam6Imen4tIRRFjWW2WYF7Q4
bYQuWqDorFLYO8ZFPxj0xthqkQ6t5eTM0EWUpRqOhanmKpmJUVnnWgABuDQvG84elfPnYWDp3Stj
6URxQ520T2ZvR9UvOHpTnYWDp3Stj6URxQbaJ7M3o+qXnD0qWrgsFy7fBxcnbtgYicvKnXmFFckY
snqEz15OTRKadMOunaEdSFKR6Sudms3NtlgsxrlhQWwrblyLeDi+bjLyrcTu8C0MfahMzn8WcZFl
DTBDbRLNJJ6FOpnQqJLbGXL1hm5aE2CyyxosGBebpWG9xccqx/nYWDp3Stj6URxQ9ttE9mb0fVcX
nD0pzsLB07pWx9KI4oNtE9mb0fVLzh6VntyebVdDcLbfl/sTGWhdmesnoCkdHJda1JxSDV0JILPV
tNDrujDnqbmZ+FsmWyy23AM3/Aku8OXZCddNU+iIXugIgIgIgIgIgIgIgIgIgIgIgIgIgIgIgIgI
gIv/Z';

```
var clickHandler = function(args) {
    if (args.item.id === 'GanttExport_pdfexport') {
        var exportProperties = {
            header: {
                fromTop: 0,
                height: 150,
                contents: [
                    {
                        type: 'Text',
                        value: 'Welcome',
                        position: { x: 380, y: 0 },
                        style: { textBrushColor: '#C25050', fontSize: 25 },
                    },
                ],
            },
        };
    }
};
```

```

        {
            type: 'Image',
            src: image,
            position: { x: 400, y: 70 },
            size: { height: 50, width: 50 },
        },
    ],
    footer: {
        fromBottom: 160,
        height: 100,
        contents: [
            {
                type: 'Text',
                value: 'Thank you!',
                position: { x: 350, y: 40 },
                style: { textBrushColor: '#C67878', fontSize: 14 }
            },
            {
                type: 'PageNumber',
                pageNumberType: 'Arabic',
                format: 'Page { $current } of { $total }',
                position: { x: 0, y: 25 },
                size: { height: 50, width: 100 },
                style: { textBrushColor: '#000000', hAlign:
'Center', vAlign: 'Bottom' }
            }
        ]
    }
};
gantChart.pdfExport(exportProperties);
}
};
var gantChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    allowPdfExport: true,
    toolbar: ['PdfExport'],
    toolbarClick: clickHandler
});
gantChart.appendTo('#GanttExport');
```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>EJ2 Gantt</title>
```

```

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Gantt Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/material.css" rel="stylesheet"
type="text/css">
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container">
    <div id="GanttExport"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Data markers in ##Platform_Name## Gantt control

Data markers are a set of events used to represent the schedule events for a task. Data markers are defined in data source as array of objects, and this value is mapped to the Gantt control using the [taskFields.indicators](#) property. You can represent more than one data marker in a task.

Data markers can be defined using the following properties:

- [date](#): Defines the date of indicator.
- [iconClass](#): Defines the icon class of indicator.
- [name](#): Defines the name of indicator.
- [tooltip](#): Defines the tooltip of indicator.

Note: Data Marker **tooltip** will be rendered only if tooltip property has value.

The following code example demonstrates how to implement data markers in the Gantt chart.

INDEX.JS

```

var ganttChart = new ej.gantt.Gantt({
  dataSource: [
    {
      TaskID: 1,
      TaskName: 'Project Initiation',
      StartDate: new Date('04/02/2019'),
      EndDate: new Date('04/21/2019'),
      subtasks: [
        {

```

```

TaskID: 2, TaskName: 'Identify Site location',
StartDate: new Date('04/02/2019'), Duration: 4, Progress: 50,
Indicators: [
    {
        'date': '04/08/2019',
        'iconClass': 'e-btn-icon e-notes-info e-icons e-
icon-left e-gantt e-notes-info::before',
        'name': 'Custom String',
        'tooltip': 'Follow up'
    },
    {
        'date': '04/11/2019',
        'iconClass': 'e-btn-icon e-notes-info e-icons e-
icon-left e-gantt e-notes-info::before',
        'name': '<span style="color:red">String
Template</span>',
    }
],
    { TaskID: 3, TaskName: 'Perform Soil test', StartDate: new
Date('04/02/2019'), Duration: 4, Progress: 50 },
    {
        TaskID: 4, TaskName: 'Soil test approval', StartDate:
new Date('04/02/2019'), Duration: 4, Progress: 50, },
    },
    {
        TaskID: 5,
        TaskName: 'Project Estimation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        subtasks: [
            {
                TaskID: 6, TaskName: 'Develop floor plan for
estimation', StartDate: new Date('04/04/2019'), Duration: 3, Progress: 50,
                Indicators: [
                    {
                        'date': '04/10/2019',
                        'iconClass': 'e-btn-icon e-notes-info e-icons e-
icon-left e-gantt e-notes-info::before',
                        'name': 'Indicator title',
                        'tooltip': 'tooltip'
                    }
                ]
            },
            { TaskID: 7, TaskName: 'List materials', StartDate: new
Date('04/04/2019'), Duration: 3, Progress: 50 },
            { TaskID: 8, TaskName: 'Estimation approval', StartDate: new
Date('04/04/2019'), Duration: 3, Progress: 50 }
        ]
    },
],
height: '450px',
taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',

```

```

        resourceInfo: 'resources',
        duration: 'Duration',
        progress: 'Progress',
        dependency: 'Predecessor',
        child: 'subtasks',
        indicators: 'Indicators'
    },
    });
    ganttChart.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <style>
    .e-gantt .e-gantt-chart .e-custom-event-marker {
      width: 1px;
      border-left: 2px green dotted;
    }
  </style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Global local in ##Platform_Name## Gantt control

Localization

The [Localization](#) library allows you to localize default text content of the Gantt. The Gantt component has static text on some features (like toolbar area text, etc.) that can be changed to other cultures (Arabic, Deutsch, French, etc.) by defining the

[locale](#) value and translation object.

The following list of properties and its values are used in the Gantt.

Locale key words | Text

emptyRecord | No records to display

id | ID

name | Name

startDate | Start Date

endDate | End Date

duration | Duration

progress | Progress

dependency | Dependency

notes | Notes

baselineStartDate | Baseline Start Date

baselineEndDate | Baseline End Date

type | Type

offset | Offset

resourceName | Resources

resourceID | Resource ID

day | day

hour | hour

minute | minute

days | days

hours | hours

minutes | minutes

generalTab | General

customTab | Custom Columns

writeNotes | Write Notes

addDialogTitle | New Task

editDialogTitle | Task Information

add | Add

edit | Edit

update | Update

delete | Delete

cancel | Cancel

search | Search

task | task

tasks | tasks

zoomIn | Zoom in

zoomOut | Zoom out

zoomToFit | Zoom to fit

expandAll | Expand all

collapseAll | Collapse all

nextTimeSpan | Next timespan

prevTimeSpan | Previous timespan

saveButton | Save

taskBeforePredecessor_FS | You moved "{0}" to start before "{1}" finishes and the two tasks are linked. As the result, the links cannot be honored. Select one action below to perform

taskAfterPredecessor_FS | You moved "{0}" away from "{1}" and the two tasks are linked. As the result, the links cannot be honored. Select one action below to perform

taskBeforePredecessor_SS | You moved "{0}" to start before "{1}" starts and the two tasks are linked. As the result, the links cannot be honored. Select one action below to perform

taskAfterPredecessor_SS | You moved "{0}" to start after "{1}" starts and the two tasks are linked. As the result, the links cannot be honored. Select one action below to perform

taskBeforePredecessor_FF | You moved "{0}" to finish before "{1}" finishes and the two tasks are linked. As the result, the links cannot be honored. Select one action below to perform

taskAfterPredecessor_FF | You moved "{0}" to finish after "{1}" finishes and the two tasks are linked. As the result, the links cannot be honored. Select one action below to perform

taskBeforePredecessor_SF | You moved "{0}" away from "{1}" to starts and the two tasks are linked. As the result, the links cannot be honored. Select one action below to perform

taskAfterPredecessor_SF | You moved "{0}" to finish after "{1}" starts and the two tasks are linked. As the result, the links cannot be honored. Select one action below to perform

okText | Ok

confirmDelete | Are you sure you want to Delete Record?

from | From

to | To

taskLink | Task Link

lag | Lag

start | Start

finish | Finish

enterValue | Enter the value

taskInformation | Task Information

deleteTask | Delete Task

deleteDependency | Delete Dependency

convert | Convert

save | Save

above | Above

below | Below

child | Child

milestone | Milestone

toTask | To Task

toMilestone | To Milestone

eventMarkers | Event markers

leftTaskLabel | Left task label

rightTaskLabel | Right task label

timelineCell | Timeline cell

confirmPredecessorDelete | Are you sure you want to remove dependency link?

taskMode | Task Mode

changeScheduleMode | Change Schedule Mode

subTasksStartDate | SubTasks Start Date

subTasksEndDate | SubTasks End Date

scheduleStartDate | Schedule Start Date

scheduleEndDate | Schedule End Date

auto | Auto

manual | Manual

zoomToFit | Zoom to fit

excelExport | Excel export

csvExport | CSV export

pdfExport | Pdf export

unit | Unit

work | Work

taskType | Task Type

unassignedTask | Unassigned Task

group | Group

Loading translations

To load translation object in an application use [load](#) function of [L10n](#) class.

The below example demonstrates the Gantt in **Deutsch** culture.

INDEX.JS

```
ej.base.L10n.load({
  'de-DE': {
    'gantt': {
      "id": "Ich würde",
      "name": "Name",
      "startDate": "Anfangsdatum",
      "duration": "Dauer",
      "progress": "Fortschritt",
    }
  }
});
var ganttChart = new ej.gantt.Gantt({
  dataSource: GanttData,
  height: '450px',
  locale: 'de-DE',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
});
ganttChart.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <style>
    .e-gantt .e-gantt-chart .e-custom-event-marker {
      width: 1px;
      border-left: 2px green dotted;
    }
  </style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Internationalization

The [Internationalization](#) library is used to globalize number, date, and time values in gantt component.

INDEX.JS

```

import * as cagregorian from './ca-gregorian.js';
import * as numbers from './numbers.js';
var GanttData = [
    {
        TaskID: 1,
        TaskName: 'Project Initiation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        subtasks: [
            { TaskID: 2, TaskName: 'Identify Site location', StartDate: new
Date('04/02/2019'), Duration: 4, Progress: 50 },
            { TaskID: 3, TaskName: 'Perform Soil test', StartDate: new
Date('04/02/2019'), Duration: 4, Progress: 50 },
            { TaskID: 4, TaskName: 'Soil test approval', StartDate: new
Date('04/02/2019'), Duration: 4, Progress: 50 },
        ]
    },
    {
        TaskID: 5,
        TaskName: 'Project Estimation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        subtasks: [
            { TaskID: 6, TaskName: 'Develop floor plan for estimation',
StartDate: new Date('04/04/2019'), Duration: 3, Progress: 50 },
            { TaskID: 7, TaskName: 'List materials', StartDate: new
Date('04/04/2019'), Duration: 3, Progress: 50 },
            { TaskID: 8, TaskName: 'Estimation approval', StartDate: new
Date('04/04/2019'), Duration: 3, Progress: 50 }
        ]
    },
];
ej.base.loadCldr(cagregorian,numbers);

```

```

ej.base.setCulture('de-DE');
ej.base.L10n.load({
    'de-DE': {
        'gantt': {
            "id": "Ich würde",
            "name": "Name",
            "startDate": "Anfangsdatum",
            "duration": "Dauer",
            "progress": "Fortschritt",
        }
    }
});
var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    locale: 'de-DE',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
});
ganttChart.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <style>
    .e-gantt .e-gantt-chart .e-custom-event-marker {
        width: 1px;
        border-left: 2px green dotted;
    }
    </style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">

```

```

        <div id="Gantt"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

* In the above sample, **Timeline** is formatted by **NumberFormatOptions** and **DateFormatOptions**.

* By default, **locale** value is **en-US**. If you want to change **en-US** culture, then set the **locale**.

Right to left (RTL)

RTL provides an option to switch the text direction and layout of the Gantt component from right to left. It improves the user experiences and accessibility for users who use right-to-left languages (Arabic, Urdu, etc.). To enable RTL Gantt, set the **enableRtl** to true.

INDEX.JS

```

ej.base.setCulture('ar-AE');
ej.base.L10n.load({
    'ar-AE': {
        "gantt": {
            "emptyRecord": "لا سجلات لعرضها",
            "id": "هوية شخصية",
            "name": "اسم",
            "startDate": "تاريخ البدء",
            "endDate": "تاريخ الانتهاء",
            "duration": "المدة الزمنية",
            "progress": "تقدم",
            "dependency": "الاعتماد",
            "notes": "ملاحظات",
            "baselineStartDate": "تاريخ البدء الأساسي",
            "baselineEndDate": "تاريخ نهاية خط الأساس",
            "taskMode": "وضع المهام",
            "changeScheduleMode": "تغيير وضع الجدول",
            "subTasksStartDate": "تاريخ بدء المهام الفرعية",
            "subTasksEndDate": "تاريخ انتهاء المهام الفرعية",
            "scheduleStartDate": "جدولة تاريخ البدء",
            "scheduleEndDate": "تاريخ انتهاء الجدول الزمني",
            "auto": "تلقائي",
            "manual": "كتيب",
            "type": "اكتب",
            "offset": "عوض",
            "resourceName": "مصادر",
            "resourceID": "معرف المورد",
            "day": "يوم",
            "hour": "ساعة",
            "minute": "دقيقة",
            "days": "أيام",
            "hours": "ساعات",
            "minutes": "الدقائق",
            "generalTab": "جنرال لواء",

```

```

"customTab": "أعمدة مخصصة",
"writeNotes": "اكتب ملاحظات",
"addDialogTitle": "مهمة جديدة",
"editDialogTitle": "معلومات المهمة",
"saveButton": "حفظ",
"add": "إضافة",
"edit": "تعديل",
"update": "تحديث",
"delete": "حذف",
"cancel": "إلغاء",
"search": "بحث",
"task": "مهمة",
"tasks": "مهام",
"zoomIn": "تكبير",
"zoomOut": "تصغير",
"zoomToFit": "تكبير لتناسب",
"excelExport": "اكسل التصدير",
"csvExport": "تصدير CSV",
"expandAll": "توسيع الكل",
"collapseAll": "انهيار جميع",
"nextTimeSpan": "الجدول الزمني التالي",
"prevTimeSpan": "الجدول الزمني السابق",
"okText": "حسنًا",
"confirmDelete": "هل أنت متأكد أنك تريد حذف السجل؟",
"from": "من عند",
"to": "إلى",
"taskLink": "رابط المهمة",
"lag": "بطئ",
"start": "بداية",
"finish": "إنهاء",
"enterValue": "أدخل القيمة",
"taskBeforePredecessor_FS": "0 '{للبدء قبل انتهاء}'  

ويتم ربط المهمتين. ونتيجة لذلك ، لا يمكن احترام الروابط. حدد إجراء '{1}'  

واحدًا أدناه للقيام به",
"taskAfterPredecessor_FS": "0 '{بعيدًا عن 1}'  

ربط المهمتين. ونتيجة لذلك ، لا يمكن احترام الروابط. حدد إجراء واحدًا أدناه  

للقيام به",
"taskBeforePredecessor_SS": "0 '{للبدء قبل أن يبدأ}'  

وربط المهمتين. ونتيجة لذلك ، لا يمكن احترام الروابط. حدد إجراء واحدًا '{1}'  

أدناه للقيام به",
"taskAfterPredecessor_SS": "0 '{للبدء بعد بدء 1}'  

وربط المهمتين. ونتيجة لذلك ، لا يمكن احترام الروابط. حدد إجراء واحدًا أدناه  

للقيام به",
"taskBeforePredecessor_FF": "0 '{للإنهاء قبل انتهاء}'  

ويتم ربط المهمتين. ونتيجة لذلك ، لا يمكن احترام الروابط. حدد إجراء '{1}'  

واحدًا أدناه للقيام به",
"taskAfterPredecessor_FF": "0 '{للإنهاء بعد انتهاء}'  

ويتم ربط المهمتين. ونتيجة لذلك ، لا يمكن احترام الروابط. حدد إجراء '{1}'  

واحدًا أدناه للقيام به",
"taskBeforePredecessor_SF": "0 '{بعيدًا عن 1}'  

التشغيل وترتبط المهمتان. ونتيجة لذلك ، لا يمكن احترام الروابط. حدد إجراء  

واحدًا أدناه للقيام به",
"taskAfterPredecessor_SF": "0 '{للإنهاء بعد بدء 1}'  

وربط المهمتين. ونتيجة لذلك ، لا يمكن احترام الروابط. حدد إجراء واحدًا أدناه  

للقيام به",
"taskInformation": "معلومات المهمة",
"deleteTask": "حذف المهمة",

```



```

        "deleteDependency": "حذف التبعية",
        "convert": "تحويل",
        "save": "حفظ",
        "above": "في الاعلى",
        "below": "أدناه",
        "child": "طفل",
        "milestone": "معلما",
        "toTask": "لمهمة",
        "toMilestone": "إلى معلم",
        "eventMarkers": "علامات الحدث",
        "leftTaskLabel": "تسمية المهمة اليسرى",
        "rightTaskLabel": "تسمية المهمة الصحيحة",
        "timelineCell": "خلية الجدول الزمني",
        "confirmPredecessorDelete": "هل أنت متأكد أنك تريد إزالة رابط",
        "التبعية؟",
        "unit": "وحدة",
        "work": "عمل",
        "taskType": "نوع المهمة",
        "unassignedTask": "مهمة غير محددة",
        "group": "مجموعة",
        "indent": "مسافة بادئة",
        "outdent": "عفا عليها الزمن",
        "segments": "شرائح",
        "splitTask": "تقسيم المهمة",
        "mergeTask": "مهمة الدمج",
        "left": "اليسار",
        "right": "حق"
    }
}
});
ej.gantt.Gantt.Inject(ej.treegrid.Toolbar);
var ganttChart = new ej.gantt.Gantt({
    dataSource: GanttData,
    locale: 'ar-AE',
    enableRtl: true,
    height: '450px',
    toolbar: ['ExpandAll', 'CollapseAll'],
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
});
ganttChart.appendTo('#Gantt');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
```

```

<link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <style>
    .e-gantt .e-gantt-chart .e-custom-event-marker {
      width: 1px;
      border-left: 2px green dotted;
    }
  </style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Internationalization](#)
- [Localization](#)

Accessibility in ##Platform_Name## Gantt control

The Gantt component followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Gantt component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support | |

| [Section 508](#) Support | |

| Screen Reader Support | |

| Right-To-Left Support | |

| Color Contrast | |

| Mobile Device Support | |

| Keyboard Navigation Support | |

| [Accessibility Checker](#) Validation | |

| [Axe-core](#) Accessibility Validation |

|

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<div> - All features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>

WAI-ARIA attributes

The Gantt component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Gantt component:

The following ARIA attributes are used in Gantt:

| Attributes | Purpose |

| --- | --- |

| **grid (role)** | This attribute is added to the **e-table** element present in the Gantt, which represents Grid part |

| **gridcell (role)** | This attribute is added to the **td** elements present within the **e-table**, which represents the work cells of Gantt |

| **columnheader (role)** | This attribute is added to the **th** elements within the **e-table**, which represents the header cells of Grid table |

| **separator (role)** | This attribute is added to the **e-split-bar** element, which represents the splitter between the Grid table and Chart |

| **dialog (role)** | This attribute is added to the **e-dialog** element, which represents the pop-up dialog |

| **toolbar (role)** | This attribute is added to the **e-gantt-toolbar** element, which represents the toolbars of Gantt |

| **aria-label** | It indicates the element's information
 It is assigned to the Gantt UI elements such as timeline cell, taskbar, left label, right label, dependency line, and event markers. |

| **aria-selected** | This attribute is assigned to the Gantt chart row, and it defaults to **false**. The value is changed to **true** when the user selects a grid cell or task |

| **aria-expanded** | This attribute is assigned to the Gantt chart parent task row. The value is changed to **true** when the user clicks a parent taskbar to expand. After the user clicked a parent taskbar to collapse, the attribute value is changed to **false** |

| **aria-grabbed** | This attribute is assigned to the taskbars of Gantt when the user tries to achieve taskbar editing |

Keyboard navigation

The Gantt component followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Gantt component.

| **Press** | **To do this** |

| --- | --- |

| **Home** | Selects the first row. |

| **End** | Selects the last row. |

| **DownArrow** | Moves the cell focus/row or cell selection downward. |

| **UpArrow** | Moves the cell focus/row or cell selection upward. |

| **LeftArrow** | Moves the cell focus/row or cell selection left side. |

| **RightArrow** | Moves the cell focus/row or cell selection right side. |

| **Ctrl + Up Arrow** | Collapses all tasks. |

| **Ctrl + Down Arrow** | Expands all tasks. |

| **Ctrl + Shift + Up Arrow** | Collapses the selected row. |

| **Ctrl + Shift + Down Arrow** | Expands the selected row. |

| **Enter** | Saves request. |

| **Esc** | Cancels request. |

| **Insert** | Adds a new row. |

| **Ctrl + Insert** | Opens addRowDialog. |

| **Ctrl + F2** | Opens editRowDialog. |

- | Delete | Deletes the selected row. |
- | Shift + F5 | FocusTask |
- | Ctrl + Shift + F | Focus search |
- | Shift + DownArrow | Extends the row/cell selection downwards. |
- | Shift + UpArrow | Extends the row/cell selection upwards. |
- | Shift + LeftArrow | Extends the cell selection to the left side. |
- | Shift + RightArrow | Extends the cell selection to the right side. |
- | Tab / Shift + Tab | To focus the close icon in the message. |
- | Alt + j | Focus Gantt component. |

Ensuring accessibility

The Gantt component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Gantt component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Gantt component with accessibility tools.

See also

- [Accessibility in Syncfusion ##Platform_Name## components](#)

Touch interaction in ##Platform_Name## Gantt control

The Gantt control supports to perform user interactions in mobile and tablet devices. This section explains how to interact with the Gantt features in touch-enabled devices.

Tooltip

To perform **touch and hold** action on a element, refer to [tooltip popup](#).

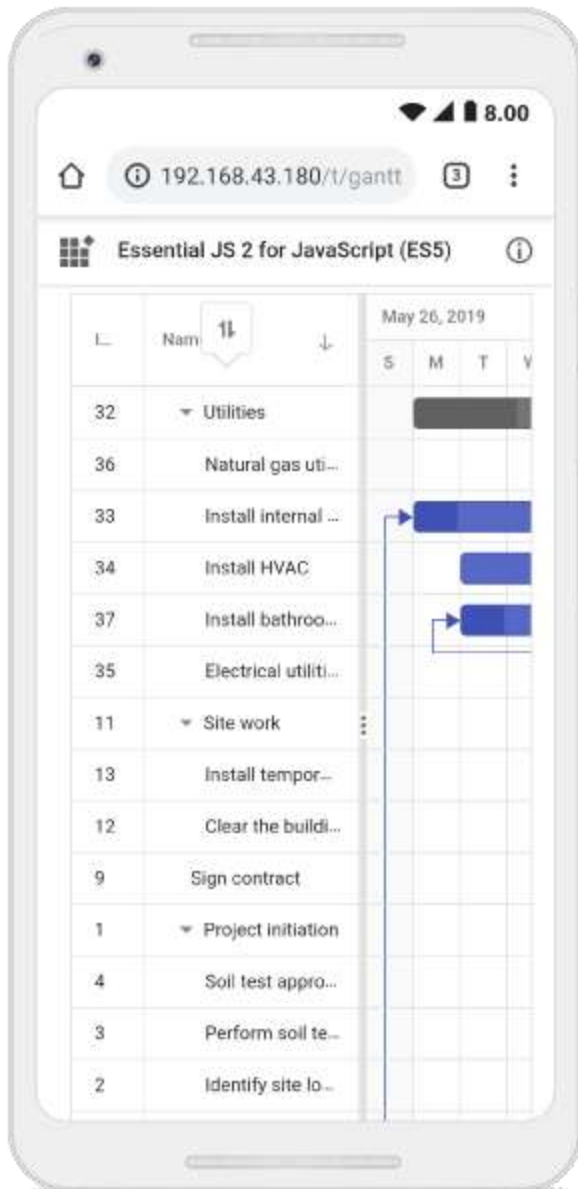
Context menu

To perform **long press** action on a row, [context menu](#) is opened, and then tap a menu item to trigger its action.

Sorting

To perform **tap** action on a column header, trigger [sorting](#) operation to the selected column. A popup is displayed for multi-column sorting. To sort multiple columns, tap the popup, and then tap the desired column headers.

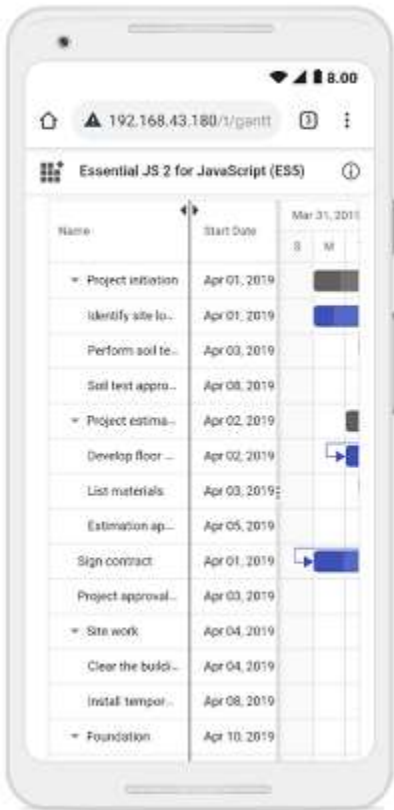
The following screenshot shows Gantt touch sorting,



Column resize

When the right edge of the column header cell is **tapped**, a floating handler will be visible over the right border of the column. To [resize](#) the column, drag the floating handler as needed.

The following screenshot represents the Gantt column resizing in touch device.



Editing

The Gantt control editing actions can be achieved using the double tap and tap and drag actions on a element.

The following table describes different types of editing modes available in Gantt.

Action | Description

Parent taskbar | You cannot create dependency relationship to parent tasks.



Taskbar without dependency | If you tap a valid child taskbar, it will create FS type dependency line between tasks, otherwise exits from task dependency edit mode.



Taskbar with dependency | If you tap the second taskbar, which has already been directly connected,



it will ask to remove it.

Removing dependency | Once you tap the taskbar with direct dependency, then confirmation dialog will be shown for removing dependency.



INDEX.JS

```
ej.gantt.Gantt.Inject(ej.gantt.Edit);
var ganttData = [
    {
        TaskID: 1,
        TaskName: 'Project Initiation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        subtasks: [
            { TaskID: 2, TaskName: 'Identify Site location',
StartDate: new Date('04/02/2019'), Duration: 3, Progress: 50 },
            { TaskID: 3, TaskName: 'Perform Soil test', StartDate:
new Date('04/02/2019'), Duration: 4, Progress: 50 },
            { TaskID: 4, TaskName: 'Soil test approval', StartDate:
new Date('04/02/2019'), Duration: 4,Predecessor:"2FS", Progress: 50 },
        ]
    },
    {
        TaskID: 5,
        TaskName: 'Project Estimation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        subtasks: [
            { TaskID: 6, TaskName: 'Develop floor plan for
estimation', StartDate: new Date('04/04/2019'), Duration: 3, Progress: 50 },
            { TaskID: 7, TaskName: 'List materials', StartDate: new
Date('04/04/2019'), Duration: 3, Progress: 50 },
            { TaskID: 8, TaskName: 'Estimation approval', StartDate:
new Date('04/04/2019'), Duration: 4,Predecessor:"6SS", Progress: 50 }
        ]
    },
];

var ganttChart = new ej.gantt.Gantt({
    dataSource: ganttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
```



```

        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        dependency: 'Predecessor',
        child: 'subtasks'
    },
    editSettings: {
        allowTaskbarEditing: true
    },
    load: function() {
        this.isAdaptive = true;
    }
});
gantChart.appendTo('#Gantt');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

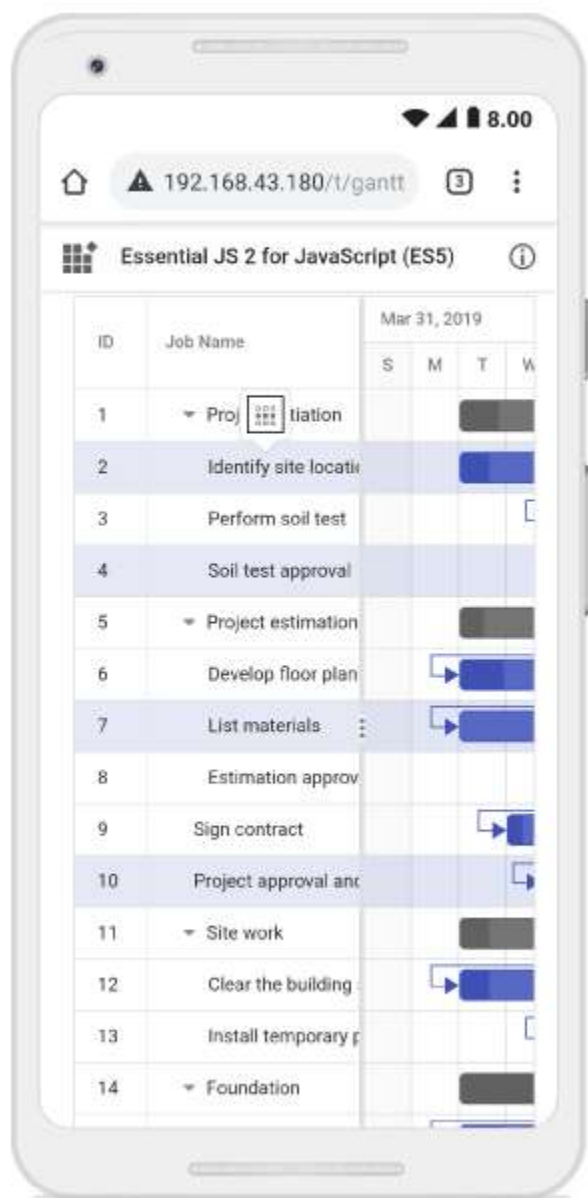
Note: In mobile device, you cannot create dependency other than FS by taskbar editing. By using cell/dialog editing, you can add all type of dependencies.

Selection

When you tap gantt row, tapped row will be selected.

[Single selection](#) : To select a single row or cell, perform single tap on it.

[Multiple selection](#) : To perform multiple selection, tap on the multiple selection popup, and then tap the desired rows or cells.



Loading animation in ##Platform_Name## Gantt control

The loading indicator is used to display a visual indicator while the Gantt is fetching data or performing certain actions, such as sorting or filtering. The gantt support two indicator types, which is achieved by setting the [loadingIndicator.indicatorType](#) property to Shimmer or Spinner. The default value of the indicator type is "Spinner."

In the following sample, the Shimmer indicator is displayed while the gantt is scrolled when using the virtual data.

INDEX.JS

```
ej.gantt.Gantt.Inject(ej.gantt.Sort,ej.gantt.Selection,ej.gantt.Filter);
```

```

var ganttChart = new ej.gantt.Gantt({
    dataSource: virtualData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        endDate: 'EndDate',
        duration: 'Duration',
        progress: 'Progress',
        parentID: 'parentID'
    },
    columns: [
        { field: 'TaskID' },
        { field: 'TaskName' },
        { field: 'StartDate' },
        { field: 'Duration' },
        { field: 'Progress' }
    ],
    loadingIndicator: { indicatorType: 'Shimmer' },
    enableVirtualization: true,
        allowFiltering: true,
        allowSorting: true
});
ganttChart.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">
    <link href="https://cdn.syncfusion.com/ej2-notifications/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {

```

```

    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Style and appearance in ##Platform_Name## Gantt control

To modify the Gantt Chart appearance, you need to override the default CSS of gantt chart. Please find the list of CSS classes and its corresponding section in Gantt Chart. Also, you have an option to create your own custom theme for all the JavaScript controls using our [Theme Studio](#).

Section | CSS Class | Purpose of Class

Root | e-gantt | This class is in the root element (div) of the gantt chart control.

Header | e-gridheader | This class is added in the root element of header element. In this class, You can override thin line between header and content of the gantt chart.

| e-table | This class is added at 'table' of the gantt chart header. This CSS class makes table width as 100 %.

| e-columnheader | This class is added at 'tr' of the gantt chart header.

Grid Content | e-gridcontent | This class is added at root of body content. This is to override background color of the body.

| e-table | This class is added to table of content. This CSS class makes table width as 100 %.

| e=row | This class is added to rows of gantt chart.

| e-altrow | This class is added to alternate rows of gantt chart. This is to override alternate row color of the gantt chart.

| e-rowcell | This class is added to all cells in the gantt chart. This is to override cells appearance and styling.

Chart Content | e-gantt-chart | This class is added to the chart side of the gantt chart.

| e-chart-row | This class is added to rows of gantt chart.

Timeline | e-timeline-header-container | This class is added to timeline of the gantt chart.

| e-header-cell-label | This class is added to the header cell of the timeline.

| e-weekend-header-cell | This class is added to the weekend cells.

Taskbar | e-taskbar-main-container | This class is added to taskbar of the gantt chart.

| e-gantt-parent-taskbar | This class is added to the parent task bar of the gantt chart.

| e-gantt-milestone | This class is added to the milestone tasks of the gantt chart.

| e-gantt-unscheduled-taskbar | This class is added to the unscheduled tasks.

| e-gantt-manualparenttaskbar | This class is added to the manual scheduled parent taskbar.

| e-gantt-child-manualtaskbar | This class is added to the manual scheduled child taskbar.

| e-gantt-unscheduled-manualtask | This class is added to the manual unscheduled tasks.

Splitter | e-split-bar | This class is added to the gantt chart splitter.

|e-resize-handler| This class is added to the resize handler of the gantt chart splitter.

|e-arrow-left| This class is added to the left arrow of the resize handler.

|e-arrow=right| This class is added to the right arrow of the resize handler.

Connector Lines|e-line| This class is added to the connector lines.

|e-connector-line-right-arrow| This class is added to the right arrow of the connector line.

|e-connector-line-left-arrow| This class is added to the left arrow of the connector line.

Labels|e-task-label| This class is added to the task labels.

|e-right-label-container| This class is added to the right label.

|e-left-label-container| This class is added to the left label.

Event Markers|e-event-markers| This class is added to the event markers.

Baseline|e-baseline-bar| This class is added to the baseline.

|e-baseline-gantt-milestone-container| This class is added to the baseline of milestone tasks.

Tooltip|e-gantt-tooltip| This class is added to the tooltip.

How To

Open add edit dialog in ##Platform_Name## Gantt control

Gantt add and edit dialogs can be opened dynamically by using [openAddDialog](#) and [openEditDialog](#) methods. The following code example shows how to open add and dialog on separate button click actions.

INDEX.JS

```
ej.gantt.Gantt.Inject(ej.gantt.Edit);
var ganttChart = new ej.gantt.Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  editSettings: {
    allowEditing: true,
    allowAdding: true
  }
});
ganttChart.appendTo('#Gantt');
var editBtn= new ej.buttons.Button();
editBtn.appendTo('#editDialog');
var addBtn= new ej.buttons.Button();
addBtn.appendTo('#addDialog');
document.getElementById('editDialog').addEventListener('click', () => {
  ganttChart.editModule.dialogModule.openEditDialog();
});
document.getElementById('addDialog').addEventListener('click', () => {
```

```
ganttChart.editModule.dialogModule.openAddDialog();
});
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <button id="editDialog">Open EditDialog</button> <button
id="addDialog">Open AddDialog</button>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

NOTE: We should select any one of the row in Gantt to open the edit dialog.

Change schedule date in ##Platform_Name## Gantt control

In the Gantt control, you can change the schedule start and end dates by clicking the custom button programmatically using the [updateProjectDates](#) method. You can pass the start and end dates as method arguments to the [updateProjectDates](#) method. You can also pass the Boolean value as an additional parameter, which is used to round-off the schedule start and end dates displayed in Gantt timeline.

INDEX.JS

```
var ganttData = [
  {
    TaskID: 1,
    TaskName: 'Project Initiation',
    StartDate: new Date('04/02/2019'),
    EndDate: new Date('04/21/2019'),
```

```

        isParent:true,
        subtasks: [
            { TaskID: 2, TaskName: 'Identify Site location', StartDate:
new Date('04/02/2019'), Duration: 3, Progress: 50,isParent:false },
            { TaskID: 3, TaskName: 'Perform Soil test', StartDate: new
Date('04/02/2019'), Duration: 3, Progress: 70, resources: [2, 3,
5],isParent:false },
            { TaskID: 4, TaskName: 'Soil test approval', StartDate: new
Date('04/02/2019'), Duration: 3, Predecessor:"2FS", Progress:
80,isParent:false },
        ]
    },
    {
        TaskID: 5,
        TaskName: 'Project Estimation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        isParent:true,
        subtasks: [
            { TaskID: 6, TaskName: 'Develop floor plan for estimation',
StartDate: new Date('04/04/2019'), Duration: 4, Progress: 50, resources:
[4],isParent:false },
            { TaskID: 7, TaskName: 'List materials', StartDate: new
Date('04/04/2019'), Duration: 4, Progress: 50, DurationUnit:'day',
resources: [4, 8],isParent:false },
            { TaskID: 8, TaskName: 'Estimation approval', StartDate: new
Date('04/04/2019'), Duration: 4,Predecessor:"6SS", DurationUnit:'minute',
Progress: 70, resources: [12, 5],isParent:false }
        ]
    },
];

var ganttChart = new ej.gantt.Gantt({
    dataSource: ganttData,
    height:'450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    }
});
ganttChart.appendTo('#Gantt');
var dateBtn= new ej.buttons.Button();
dateBtn.appendTo('#updateSchedule');
document.getElementById('updateSchedule').addEventListener('click',
function() {
    ganttChart.updateProjectDates(new Date('01/10/2019'),new
Date('06/20/2019'),true);
});

```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
```

```

<title>EJ2 Gantt</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Gantt Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <button id="updateSchedule">Change Schedule Dates</button>
    <div id="container">
      <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Copy paste records in ##Platform_Name## Gantt control

You can copy and paste a record in the Gantt chart by using the `addRecord` method and `custom context menu`. It is also possible to copy and paste the parent record with multiple hierarchical child records on the required position.

INDEX.JS

```

ej.base.enableRipple(true);
var copiedRecord;
var enableFlag;
var gantt = new ej.gantt.Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    dependency: 'Predecessor',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  editSettings: {
    allowAdding: true,

```



```

        allowEditing: true,
        allowDeleting: true,
        allowTaskbarEditing: true
    },
    toolbar: ['Add', 'Edit', 'Update', 'Delete', 'Cancel', 'ExpandAll',
    'CollapseAll'],
    allowSelection: true,
    enableContextMenu: true,
    contextMenuItems: [
        { text: 'Copy', target: '.e-content', id: 'copy' },
        { text: 'Paste', target: '.e-content', id: 'paste' },
    ],
    contextMenuClick: function (args) {
        if (args.item.id === 'copy') {
            copiedRecord = args.rowData;
            copiedRecord.taskData.TaskID = gantt.currentViewData.length + 1;
        }
        if (args.item.id === 'paste') {
            gantt.addRecord(copiedRecord.taskData, 'Below', args.rowData.index);
            if (copiedRecord.hasChildRecords) {
                addChildRecords(copiedRecord, args.rowData.index + 1);
            }
            copiedRecord = undefined;
        }
    },
    contextMenuOpen: function (args) {
        if (args.type !== 'Header') {
            if (copiedRecord) {
                args.hideItems.push('Copy');
            } else {
                args.hideItems.push('Paste');
            }
        }
    }
});
gantt.appendTo('#Gantt');
function addChildRecords(record, index) {
    for(var i=0; i<record.childRecords.length; i++) {
        var childRecord = record.childRecords[i];
        childRecord.taskData.TaskID = ganttChart.currentViewData.length
+ 1;
        ganttChart.addRecord(childRecord.taskData, 'Child', index);
        if (childRecord.hasChildRecords) {
            addChildRecords(childRecord, index + (i+1));
        }
    }
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">

```

```

<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Maintain record index in ##Platform_Name## Gantt control

Row dropped record's index position can be maintained in the Gantt chart by changing the database table index position using the `rowDrop` event. In this event, the `fromIndex` and `dropIndex` values can be passed to the server side using Ajax request. On the server side, the `insert` and `insertAtTop` methods are used to update the row index position. The following code snippets explain the solution.

`ts

```

import { Gantt, Selection, Edit, RowDD, IGanttData } from '../src/index';
import { Ajax } from '@syncfusion/ej2-base';

Gantt.Inject(Selection, Edit, RowDD);

let gantt: Gantt = new Gantt({
  dataSource: new DataManager({
    url: 'https://localhost:44379/Home/UrlDatasource',
    adaptor: new UrlAdaptor,
    crossDomain: true,
    batchUrl: 'https://localhost:44379/Home/BatchUpdate'
  }),
  taskFields: {
    id: 'TaskID',

```

```

name: 'TaskName',
startDate: 'StartDate',
duration: 'Duration',
progress: 'Progress',
dependency: 'Predecessor',
child: 'subtasks'
},
allowRowDragAndDrop: true,
rowDrop: function (args) {
let record = this.flatData[args.fromIndex][this.taskFields.id];
let record2 = this.flatData[args.dropIndex][this.taskFields.id];
let data: IGanttData = args.data[0];
let uri = 'https://localhost:44379/Home/RowDropMethod';
let dragdropdata = {
record: data[0].taskData,
position: args.dropIndex,
dragidMapping: record,
dropidMapping: record2
};
let ajax = new Ajax(
{
url: uri,
type: 'POST',
contentType: "application/json",
data: JSON.stringify(dragdropdata),
});
ajax.send();
}
});
gantt.appendTo('#ganttContainer');
`ts
public IActionResult RowDropMethod([FromBody] DragDropData value)

```

```
{
var data = new CRUDModel();
copyRecord = true;
if (value.position == "bottomSegment" || value.position == "topSegment")
{
{
var childCount = 0;
int parent1 = (int)value.record.parentID;
childCount += FindChildRecords(parent1);
if (childCount == 1)
{
var i = 0;
for (; i < DataList.Count; i++)
{
if (DataList[i].taskID == parent1)
{
DataList[i].isParent = false;
break;
}
if (DataList[i].taskID == value.record.taskID)
{
DataList[i].parentID = null;
break;
}
}
}
DataList.Remove(DataList.Where(ds => ds.taskID == value.dragidMapping).FirstOrDefault());
var j = 0;
for (; j < DataList.Count; j++)
{
if (DataList[j].taskID == value.dropidMapping)
{

```

```

value.record.parentID = DataList[j].parentID;
break;
}
}
data.Value = value.record;
if (value.position == "bottomSegment")
{
this.Insert(data, value.dropidMapping);
}
else if (value.position == "topSegment")
{
this.InsertAtTop(data, value.dropidMapping);
}
}
else if (value.position == "middleSegment")
{
DataList.Remove(DataList.Where(ds => ds.taskID == value.dragidMapping).FirstOrDefault());
value.record.parentID = value.dropidMapping;
FindDropdata(value.dropidMapping);
data.Value = value.record;
this.Insert(data, value.dropidMapping);
}
copyRecord = false;
return Json(new { updatedRecords = value.record });
}
`

```

Custom field in ##Platform_Name## Gantt control

Generally in Gantt, Custom fields are displayed in the Custom Tab of the Add/Edit dialogs. However, they can be included in the General Tab of Add/Edit Dialog Box using `actionBegin` and `actionComplete` events. These events are used to append the custom field to the dialog box. The following code snippets demonstrate the solution.

INDEX.JS

```

var divElement;
var inputs = {
    booleanedit: ej.buttons.CheckBox,
    dropdownedit: ej.dropdowns.DropDownList,

```

```

datepickeredit: ej.calendars.DatePicker,
datetimepickeredit: ej.calendars.DateTimePicker,
maskededit: ej.inputs.MaskedTextBox,
numericedit: ej.inputs.NumericTextBox,
stringedit: ej.inputs.TextBox
};
var gantt = new ej.gantt.Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks',
    },
    editSettings: {
        allowAdding: true,
        allowEditing: true,
        allowDeleting: true,
        mode: "Auto"
    },
    columns: [
        { field: 'TaskID', width: '150' },
        { field: 'TaskName', width: '250' },
        { field: 'StartDate', width: '250' },
        { field: 'Duration', width: '250' },
        { field: 'Progress', width: '250' },
        { field: 'CustomField', width: '250' }
    ],
    toolbar: ['Add', 'Cancel', 'CollapseAll', 'Delete', 'Edit', 'ExpandAll',
'Update'],
    editDialogFields: [
        { type: 'General', headerText: 'General' },
        { type: 'Dependency' },
        { type: 'Resources' },
        { type: 'Notes' }
    ],
    addDialogFields: [
        { type: 'General', headerText: 'General' },
        { type: 'Dependency' },
        { type: 'Resources' },
        { type: 'Notes' }
    ],
    actionBegin: function(args) {
        if (args.requestType === "beforeOpenEditDialog" || args.requestType
=== "beforeOpenAddDialog" ) {
            var column = this.columnByField("CustomField");
            divElement = this.createElement("div", {
                className: "e-edit-form-column"
            });
            var inputElement;
            inputElement = this.createElement("input", {
                attrs: {
                    type: "text",
                    id: this.controlId + "" + column.field,

```

```

        name: column.field,
        title: column.field
    }
});
divElement.appendChild(inputElement);
var input = {
    enabled: true,
    floatLabelType: "Auto",
    placeholder: "CustomField",
    value: args.rowData.CustomField
};
var inputObj = new inputs[column.editType](input);
inputObj.appendTo(inputElement);
    }
},
actionComplete: function(args) {
    if (args.requestType === "openEditDialog" || args.requestType ===
"openAddDialog") {
        var generalTab = document.getElementById(
            this.controlId + "GeneralTabContainer"
        );
        generalTab.appendChild(divElement);
    }
}
});
ganttt.appendTo('#Gantt');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
```

```

}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Maintain zoom to fit in ##Platform_Name## Gantt control

In the Gantt control, While performing edit actions or dynamically change dataSource, the timeline gets refreshed. When zoomToFit toolbar item is clicked and perform editing actions or dynamically change dataSource, the timeline gets refreshed. So that, the timeline will not fit to the project any more.

Maintain zoomToFit after edit actions

We can maintain zoomToFit after editing actions(cell edit,dialog edit,taskbar edit) by using [fitToProject](#) method in `actionComplete` and `taskbarEdited` event.

INDEX.JS

```

var ganttChart = new ej.gantt.Gantt({
  dataSource: GanttData,
  allowSelection: true,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    dependency: 'Predecessor',
    child: 'subtasks'
  },
  toolbar: ['Edit', 'ZoomToFit'],
  editSettings: {
    allowEditing: true,
    allowTaskbarEditing: true,
  },
  labelSettings: {
    leftLabel: 'TaskName'
  },
  projectStartDate: new Date('03/24/2019'),
  projectEndDate: new Date('04/28/2019'),
  taskbarEdited: function(args) {
    if (args) {
      var obj = document.getElementById("Gantt").ej2_instances[0];
      obj.dataSource = GanttData;
      obj.fitToProject();
    }
  },
  actionComplete: function (args) {
    if ((args.action === "CellEditing" || args.action === "DialogEditing")
    && args.requestType === "save") {
      var obj = document.getElementById("Gantt").ej2_instances[0];
      obj.dataSource = GanttData;
      obj.fitToProject();
    }
  }
});

```



```
ganttChart.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Maintain zoomToFit after change dataSource dynamically

We can maintain **zoomToFit** after change dataSource dynamically, by calling [fitToProject](#) method in dataBound event.

INDEX.JS

```
var ganttChart = new ej.gantt.Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    dependency: 'Predecessor',
    child: 'subtasks'
  },
},
```

```

        toolbar: ['ZoomToFit'],
        labelSettings: {
            leftLabel: 'TaskName'
        },
        projectStartDate: new Date('03/24/2019'),
        projectEndDate: new Date('04/28/2019'),
        dataBound: function (args) {
            this.fitToProject();
        }
    });
    ganttChart.appendTo('#Gantt');
    document.getElementById('changeData').addEventListener('click', function ()
    {
        var obj = document.getElementById('Gantt').ej2_instances[0];
        obj.dataSource = data;
    });

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="changeData">Change Data</button>
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Drag and drop in ##Platform_Name## Gantt control

In Gantt, it is possible to drag a record from another component and drop it in Gantt chart with updating the Gantt record. Here, dragging an item from **TreeView** component to Gantt and that item is updated as a resource for the Gantt record, we can achieve this, by using [nodeDragStop](#) event of **TreeView** control.

INDEX.JS

```
ej.gantt.Gantt.Inject(ej.gantt.Edit);
var treeObj = new ej.navigations.TreeView({
    fields: { dataSource: editingResources, id: 'resourceId', text:
    'resourceName' },
    allowDragAndDrop: true,
    height: "200px",
    nodeDragStop: function (args) {
        var chartEle = ej.base.closest(args.target, '.e-chart-row');
        var gridEle = ej.base.closest(args.target, '.e-row');
        if(gridEle) {
            var index = ganttChart.treeGrid.getRows().indexOf(gridEle);
            ganttChart.selectRow(index);
        }
        if (chartEle) {
            var index = chartEle.ariaRowIndex;
            ganttChart.selectRow(Number(index));
        }
        var record= args.draggedNodeData;
        var selectedData = ganttChart.flatData[ganttChart.selectedRowIndex];
        var selectedDataResource = selectedData.taskData.resources;
        var resources = [];
        if (selectedDataResource) {
            for (var i = 0; i < selectedDataResource.length; i++) {
                resources.push(selectedDataResource[i].resourceId);
            }
        }
        resources.push(parseInt(record.id));
        if (chartEle || gridEle) {
            var data = {
                TaskID: selectedData.taskData.TaskID,
                resources: resources
            };
            ganttChart.updateRecordByID(data);
            var elements = document.querySelectorAll('.e-drag-item');
            while (elements.length > 0 && elements[0].parentNode) {
                elements[0].parentNode.removeChild(elements[0]);
            }
        }
    }
});
var ganttChart = new ej.gantt.Gantt({
    dataSource: editingData,
    resources: editingResources,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
```

```

        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks',
        dependency: 'Predecessor',
        resourceInfo: 'resources'
    },
    editSettings: {
        allowEditing: true
    },
    resourceFields: {
        id: 'resourceId',
        name: 'resourceName'
    },
    splitterSettings: {
        position: "30%"
    },
    labelSettings: {
        rightLabel: 'resources'
    }
});
gantChart.appendTo('#Gantt');
treeObj.appendTo('#TreeView');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <b><label>Gantt</label></b>
        <div id="Gantt"></div>
        <b><label>List</label></b>
        <div id="TreeView"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";

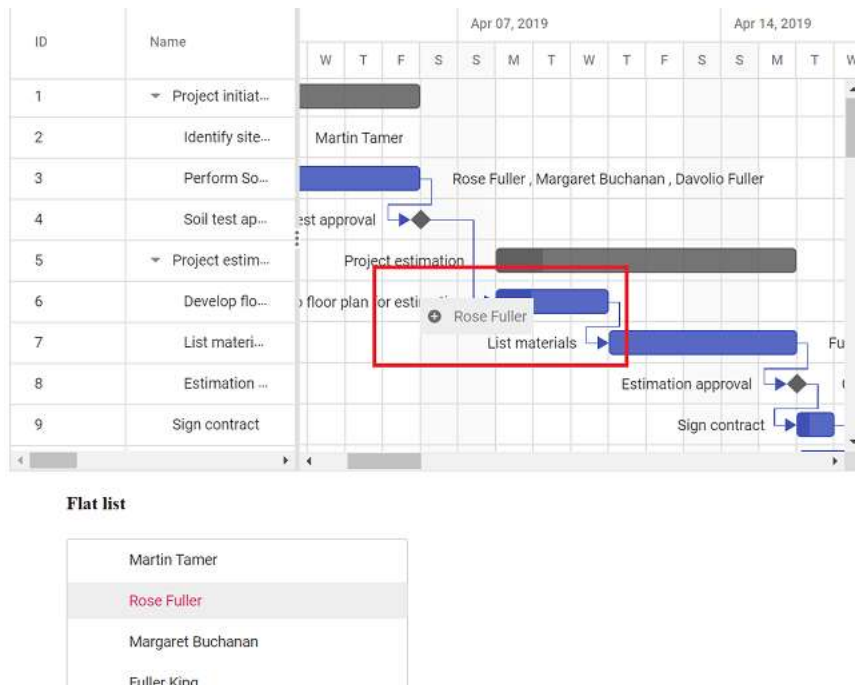
```

```

}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The following screenshot shows dropping record from another component in to Gantt, and **Rose Fuller** is added as resource for the task **Develop floor plan estimation**.



New row position in ##Platform_Name## Gantt control

In Gantt, a new row can be added in one of the following positions: Top, Bottom, Above, Below and Child. This position can be specified through the `newRowPosition` property. We can make use of the `toolbarClick` event to create a context menu that specifies the position in which the new row is to be added when adding a record through toolbar click.

The following code snippets demonstrate how to achieve this.

INDEX.JS

```

var clickHandler = function(args) {
    if (args.item.id === 'GanttExport_add') {
        var contextMenuObj =
document.getElementById("contextmenu").ej2_instances[0];
        contextMenuObj.open(60, 20);
    }
};
ej.gantt.Gantt.Inject(ej.gantt.Toolbar);
var menuItems = [
    {
        text: 'Bottom'
    },
    {
        text: 'Above'
    },

```

```

        {
            text: 'Below'
        },
        {
            text: 'Child'
        },
        {
            text: 'Top'
        }
    ]];
    var menuOptions = {
        items: menuItems,
        select: select
    };
    var menuObj = new ej.navigations.ContextMenu(menuOptions, '#contextmenu');
    function select(args) {
        let gantt = (document.getElementsByClassName('e-gantt')[0]).ej2_instances[0];
        if (args.item.text === "Bottom") {
            gantt.editSettings.newRowPosition = "Bottom";
            gantt.openAddDialog();
        } else if (args.item.text === "Above") {
            if (gantt.selectedRowIndex == -1) {
                alert("Please select any row");
            } else {
                gantt.editSettings.newRowPosition = "Above";
                gantt.openAddDialog();
            }
        } else if (args.item.text === "Below") {
            if (gantt.selectedRowIndex == -1) {
                alert("Please select any row");
            } else {
                gantt.editSettings.newRowPosition = "Below";
                gantt.openAddDialog();
            }
        } else if (args.item.text === "Child") {
            if (gantt.selectedRowIndex == -1) {
                alert("Please select any row");
            } else {
                gantt.editSettings.newRowPosition = "Child";
                gantt.openAddDialog();
            }
        } else if (args.item.text === "Top") {
            gantt.editSettings.newRowPosition = "Top";
            gantt.openAddDialog();
        }
    }
    var ganttChart = new ej.gantt.Gantt({
        dataSource: GanttData,
        height: '450px',
        allowSelection: true,
        taskFields: {
            id: 'TaskID',
            name: 'TaskName',
            startDate: 'StartDate',
            duration: 'Duration',
            progress: 'Progress',
            child: 'subtasks'
        }
    });

```

```

    },
    editSettings: {
        allowAdding: true
    },
    toolbar: ['Edit', { text: 'Add', tooltipText: 'Add', id: 'Add' }],
    toolbarClick: clickHandler,
});
gantChart.appendTo('#Gantt');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
    <ul id="contextmenu"></ul>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Render timeline from 1 to 365 days in ##Platform_Name## Gantt control

Gantt chart contains different types of in-built timeline view modes and it can be defined by using [timelineViewMode](#) property and also we can customize the timescale mode of top tier and bottom tier by using [topTier.unit](#) and [bottomTier.unit](#) properties. Timeline tier's unit can be defined by using [unit](#) property and [format](#) property was used to define date format of timeline cell and [formatter](#) property was used to define custom method to format the date value of timeline cell.

In the [bottomTier.unit](#) timescale mode, it is possible to display timeline from 1 to 365 days by making use of the formatter in the `timelineSettings` property. The following example shows how to use the formatter method for timeline cells.

INDEX.JS

```

var GanttData = [
    {
        TaskID: 1,
        TaskName: 'Product concept',
        StartDate: new Date('01/02/2019'),
        EndDate: new Date('01/21/2019'),
        subtasks: [
            { TaskID: 2, TaskName: 'Defining the product and its usage',
              StartDate: new Date('01/02/2019'), Duration: 3, Progress: 30 },
            { TaskID: 3, TaskName: 'Defining target audience', StartDate:
              new Date('01/02/2019'), Duration: 3 },
            {
                TaskID: 4, TaskName: 'Prepare product sketch and notes',
                StartDate: new Date('01/02/2019'), Duration: 2,
                Predecessor: '2', Progress: 30
            },
        ],
    },
    {
        TaskID: 5,
        TaskName: 'Market research',
        StartDate: new Date('01/02/2019'),
        EndDate: new Date('01/21/2019'),
        subtasks: [
            {
                TaskID: 6,
                TaskName: 'Demand analysis',
                StartDate: new Date('01/04/2019'),
                EndDate: new Date('01/21/2019'),
                subtasks: [
                    {
                        TaskID: 7, TaskName: 'Customer strength', StartDate:
                          new Date('01/04/2019'), Duration: 4,
                        Predecessor: '5', Progress: 30
                    },
                    { TaskID: 8, TaskName: 'Market opportunity analysis',
                      StartDate: new Date('01/04/2019'), Duration: 4, Predecessor: '5' }
                ]
            },
            {
                TaskID: 9, TaskName: 'Competitor analysis', StartDate: new
                  Date('01/04/2019'), Duration: 4,
                Predecessor: '7, 8', Progress: 30
            },
            { TaskID: 10, TaskName: 'Product strength analysis', StartDate:
              new Date('01/04/2019'), Duration: 4, Predecessor: '9' },
            {
                TaskID: 11, TaskName: 'Research complete', StartDate: new
                  Date('01/04/2019'), Duration: 0, Predecessor: '10',
                Indicators: [
                    {
                        'date': '01/20/2019',
                        'name': 'Research completed',
                        'tooltip': 'Research completed',
                        'iconClass': 'description e-icons'
                    }
                ]
            }
        ]
    }
]

```



```

    }
    ]
  }
  ],
},
];
var gantt = new ej.gantt.Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  timelineSettings: {
    timelineUnitSize: 50,
    topTier: {
      unit: "Month",
      format: "MMM dd, y"
    },
    bottomTier: {
      unit: "Day",
      formatter: date => {
        let presentDate = new Date(
          date.getFullYear(),
          date.getMonth(),
          date.getDate()
        );
        var start = new Date(presentDate.getFullYear(), 0, 0);
        var diff = Number(presentDate) - Number(start);
        var oneDay = 1000 * 60 * 60 * 24;
        var day = Math.floor(diff / oneDay);
        return day;
      }
    }
  },
  projectStartDate: new Date("01/01/2019"),
  projectEndDate: new Date("01/01/2020")
});
gantt.appendTo('#Gantt');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Grid

Getting started in ##Platform_Name## Grid control

This section explains the steps to create a simple Grid and demonstrates the basic usage of the grid component using the Essential JS 2

[quickstart](#) seed repository. This seed repository is pre-configured with the Essential JS 2 package.

Dependencies

Following is the list of minimum dependencies required to use the grid.

```

|-- @syncfusion/ej2-grids
|-- @syncfusion/ej2-base
|-- @syncfusion/ej2-data
|-- @syncfusion/ej2-excel-export
|-- @syncfusion/ej2-file-utils
|-- @syncfusion/ej2-compression
|-- @syncfusion/ej2-pdf-export
|-- @syncfusion/ej2-file-utils
|-- @syncfusion/ej2-compression
,

```

Setup for local development

Clone the Essential JS 2 quickstart application project from [GitHub](#), and install the necessary npm packages using the following command line scripts.

```
`  
git clone https://github.com/syncfusion/ej2-quickstart.git quickstart  
cd quickstart  
npm install  
`
```

Configuring system JS

Syncfusion Grid packages have to be mapped in the **system.config.js** configuration file.

```
System.config({  
  paths: {  
    'npm:': './node_modules/',  
    'syncfusion:': 'npm:@syncfusion/'  
  },  
  map: {  
    app: 'app',  
    //Syncfusion packages mapping  
    "@syncfusion/ej2-base": "syncfusion:ej2-base/dist/ej2-base.umd.min.js",  
    "@syncfusion/ej2-data": "syncfusion:ej2-data/dist/ej2-data.umd.min.js",  
    "@syncfusion/ej2-buttons": "syncfusion:ej2-buttons/dist/ej2-buttons.umd.min.js",  
    "@syncfusion/ej2-splitbuttons": "syncfusion:ej2-splitbuttons/dist/ej2-splitbuttons.umd.min.js",  
    "@syncfusion/ej2-popups": "syncfusion:ej2-popups/dist/ej2-popups.umd.min.js",  
    "@syncfusion/ej2-navigations": "syncfusion:ej2-navigations/dist/ej2-navigations.umd.min.js",  
    "@syncfusion/ej2-inputs": "syncfusion:ej2-inputs/dist/ej2-inputs.umd.min.js",  
    "@syncfusion/ej2-dropdowns": "syncfusion:ej2-dropdowns/dist/ej2-dropdowns.umd.min.js",  
    "@syncfusion/ej2-notifications": "syncfusion:ej2-notifications/dist/ej2-notifications.umd.min.js",  
    "@syncfusion/ej2-calendars": "syncfusion:ej2-calendars/dist/ej2-calendars.umd.min.js",  
    "@syncfusion/ej2-lists": "syncfusion:ej2-lists/dist/ej2-lists.umd.min.js",  
    "@syncfusion/ej2-grids": "syncfusion:ej2-grids/dist/ej2-grids.umd.min.js",  
    "@syncfusion/ej2-excel-export": "syncfusion:ej2-excel-export/dist/ej2-excel-export.umd.min.js",  
    "@syncfusion/ej2-pdf-export": "syncfusion:ej2-pdf-export/dist/ej2-pdf-export.umd.min.js",  
    "@syncfusion/ej2-file-utils": "syncfusion:ej2-file-utils/dist/ej2-file-utils.umd.min.js",  
    "@syncfusion/ej2-querybuilder": "syncfusion:ej2-querybuilder/dist/ej2-querybuilder.umd.min.js",  
    "@syncfusion/ej2-compression": "syncfusion:ej2-compression/dist/ej2-compression.umd.min.js"  
  },  
});
```

```
packages: {  
  'app': { main: 'app', defaultExtension: 'js' }  
}  
});  
System.import('app');  
`
```

Adding CSS reference

Combined CSS files are available in the Essential JS 2 package root folder. This can be referenced in your `[src/styles/styles.css]` using the following code.

```
`  
@import '../node_modules/@syncfusion/ej2/material.css';  
`
```

To know about individual component CSS, please refer to [Individual Component theme files](#) section.

Adding Grid component

You can start adding Essential JS 2 grid component to the application. To get started, add the grid component in **app.ts** and **index.html** files using the following code.

Place the following grid code in the **app.ts**.

```
`ts  
import { Grid } from '@syncfusion/ej2-grids';  
import { data } from './datasource';  
let grid: Grid = new Grid({  
  dataSource: data,  
  columns: [  
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right', width: 120, type: 'number' },  
    { field: 'CustomerID', width: 140, headerText: 'Customer ID', type: 'string' },  
    { field: 'Freight', headerText: 'Freight', textAlign: 'Right', width: 120, format: 'C' },  
    { field: 'OrderDate', headerText: 'Order Date', width: 140, format: 'yMd' }  
  ],  
  height: 315  
});  
grid.appendTo('#Grid');  
`
```

Now, add an HTML div element to act as the grid element in **index.html** using the following code.

```
`html
```

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>Essential JS 2</title>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-scale=1.0, user-scalable=no" />
<meta name="description" content="Essential JS 2" />
<meta name="author" content="Syncfusion" />
<link rel="shortcut icon" href="resources/favicon.ico" />
<link href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css" rel="stylesheet" />
<!--style reference from app-->
<link href="/styles/styles.css" rel="stylesheet" />
<!--system js reference and configuration-->
</head>
<body>
<!--Element which will render as Grid-->
<div id="Grid"></div>
</body>
</html>
```

Module injection

To create grids with additional features, inject the required modules. The following modules are used to extend grid's basic functionality.

- [Page](#) - Inject this module to use paging feature.
- [Sort](#) - Inject this module to use sorting feature.
- [Filter](#) - Inject this module to use filtering feature.
- [Group](#) - Inject this module to use grouping feature.
- **ExcelExport** - Inject this module to use Excel export feature.
- **PdfExport** - Inject this module to use PDF export feature.

These modules should be injected into the grid using the **Grid.Inject** method.

Additional feature modules are available [here](#).

Enable paging

The paging feature enables users to view the grid record in a paged view. It can be enabled by setting the [allowPaging](#) property to true. Inject the [Page](#) module as follows. If the [Page](#) module is not injected, the pager will not be rendered in the grid. Pager can be customized using the [pageSettings](#) property.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Page);
var grid = new ej.grids.Grid({
  dataSource: data,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign:
'Right', width: 120, type: 'number' },
    { field: 'CustomerID', width: 140, headerText: 'Customer
ID', type: 'string' },
    { field: 'Freight', headerText: 'Freight', textAlign:
'Right', width: 120, format: 'C' },
    { field: 'OrderDate', headerText: 'Order Date', width: 140,
format: 'yMd' }
  ],
  allowPaging: true,
  pageSettings: { pageSize: 7 }
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <style>

```

```

        .e-row[aria-selected="true"] .e-customizedExpandcell {
            background-color: #e0e0e0;
        }
        .e-grid.e-gridhover tr[role='row']:hover {
            background-color: #eee;
        }
        .e-expand::before {
            content: '\e5b8';
        }
        .empImage {
            margin: 6px 16px;
            float: left;
            width: 50px;
            height: 50px;
        }
    </style>
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
    </head>
    <body>

        <div id="container">
            <div id="Grid"></div>
        </div>
    <script>
    var ele = document.getElementById('container');
    if(ele) {
        ele.style.visibility = "visible";
    }
    </script>
    <script src="index.js" type="text/javascript"></script>
    </body></html>

```

Enable sorting

The sorting feature enables you to order the records. It can be enabled by setting the [allowSorting](#) property as true. Inject the [Sort](#)

module as follows. If [Sort](#) module is not injected, you cannot sort when a header is clicked. Sorting feature can be customized using the [sortSettings](#) property.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Page, ej.grids.Sort);
var grid = new ej.grids.Grid({
    dataSource: data,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign:
'Right', width: 120, type: 'number' },
        { field: 'CustomerID', width: 140, headerText: 'Customer
ID', type: 'string' },
        { field: 'Freight', headerText: 'Freight', textAlign:
'Right', width: 120, format: 'C' },

```

```

        { field: 'OrderDate', headerText: 'Order Date', width: 140,
format: 'yMd' }
    ],
    allowSorting: true,
    allowPaging: true,
    pageSettings: { pageSize: 7 }
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <style>
        .e-row[aria-selected="true"] .e-customizedExpandcell {
            background-color: #e0e0e0;
        }
        .e-grid.e-gridhover tr[role='row']:hover {
            background-color: #eee;
        }
        .e-expand::before {
            content: '\e5b8';
        }
        .empImage {
            margin: 6px 16px;
        }
    </style>

```



```

        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Enable filtering

The filtering feature enables you to view reduced amount of records based on filter criteria. It can be enabled by setting the [allowFiltering](#) property as true. The [Filter](#) module has to be injected as follows.

If [Filter](#) module is not injected, filter bar will not be rendered in the grid. Filtering feature can be customized using the [filterSettings](#) property.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Page, ej.grids.Sort, ej.grids.Filter);
var grid = new ej.grids.Grid({
    dataSource: data,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign:
'Right', width: 120, type: 'number' },
        { field: 'CustomerID', width: 140, headerText: 'Customer
ID', type: 'string' },
        { field: 'Freight', headerText: 'Freight', textAlign:
'Right', width: 120, format: 'C' },
        { field: 'OrderDate', headerText: 'Order Date', width: 140,
format: 'yMd' }
    ],
    allowFiltering: true,
    allowPaging: true,
    pageSettings: { pageSize: 6 },
    allowSorting: true
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
      background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
      background-color: #eee;
    }
    .e-expand::before {
      content: '\e5b8';
    }
    .empImage {
      margin: 6px 16px;
      float: left;
      width: 50px;
      height: 50px;
    }
  </style>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```

```

    <div id="container">
      <div id="Grid"></div>
    </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Enable grouping

The grouping feature enables users to view the grid record in a grouped view. It can be enabled by setting the [allowGrouping](#) property to true. The [Group](#) module has to be injected as follows. If [Group](#) module is not injected, the group drop area will not be rendered in the grid. Grouping feature can be customized using the [groupSettings](#) property.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Page, ej.grids.Sort, ej.grids.Filter,
ej.grids.Group);
var grid = new ej.grids.Grid({
  dataSource: data,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign:
'Right', width: 120, type: 'number' },
    { field: 'CustomerID', width: 140, headerText: 'Customer
ID', type: 'string' },
    { field: 'Freight', headerText: 'Freight', textAlign:
'Right', width: 120, format: 'C' },
    { field: 'OrderDate', headerText: 'Order Date', width: 140,
format: 'yMd' }
  ],
  height: 180,
  pageSettings: { pageSize: 7 },
  allowGrouping: true,
  allowPaging: true,
  allowSorting: true,
  allowFiltering: true
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}

```

```

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Run the application

The quickstart project is configured to compile and run the application in the browser. Use the following command to run the application.

`

npm start

`

Output will be displayed as follows.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Page, ej.grids.Sort, ej.grids.Filter,
ej.grids.Group);
var grid = new ej.grids.Grid({
  dataSource: data,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign:
'Right', width: 120, type: 'number' },
    { field: 'CustomerID', width: 140, headerText: 'Customer
ID', type: 'string' },
    { field: 'Freight', headerText: 'Freight', textAlign:
'Right', width: 120, format: 'C' },
    { field: 'OrderDate', headerText: 'Order Date', width: 140,
format: 'yMd' }
  ],
  height: 175,
  pageSettings: { pageSize: 7 },
  allowGrouping: true,
  allowPaging: true,
  allowSorting: true,
  allowFiltering: true
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Overview of Grid](#)
- [How to open pdf document on button click inside a Grid](#)
- [How to disable the default keyboard actions in Grid](#)

Module in ##Platform_Name## Grid control

The following feature modules should be injected to extend grid's functionality.

The available grid modules are:

Module	Description
-----	-----
Page	Inject this module to use paging feature.
Sort	Inject this module to use sorting feature.
Filter	Inject this module to use filtering feature.
Group	Inject this module to use grouping feature.
Edit	Inject this module to use editing feature.
Aggregate	Inject this module to use aggregate feature.
ColumnChooser	Inject this module to use column chooser feature.
ColumnMenu	Inject this module to use column menu feature.
CommandColumn	Inject this module to use command column feature.
ContextMenu	Inject this module to use context menu feature.
DetailRow	Inject this module to use detail template feature.
ForeignKey	Inject this module to use foreign key feature.
Freeze	Inject this module to use frozen rows and columns feature.
Resize	Inject this module to use resize feature.
Reorder	Inject this module to use reorder feature.
RowDD	Inject this module to use row drag and drop feature.
Search	Inject this module to use search feature and this is a default injected module.
Selection	Inject this module to use selection feature and this is a default injected module.
Scroll	Inject this module to use scrolling feature and this is a default injected module.
Print	Inject this module to use to use print feature and this is a default injected module.
Toolbar	Inject this module to use toolbar feature.
VirtualScroll	Inject this module to use virtual scroll feature.

| **ExcelExport** | Inject this module to use excel export feature. |

| **PdfExport** | Inject this module to use PDF export feature. |

These modules should be injected into the grid using the **Grid.Inject** method.

Data Binding

Data binding in ##Platform_Name## Grid control

The Grid uses [DataManager](#), which supports both RESTful JSON data services binding and local JavaScript object array binding. The [dataSource](#) property can be assigned either with the instance of [DataManager](#) or JavaScript object array collection.

It supports two kinds of data binding method:

- Local data
- Remote data

Sending additional parameters to the server

To add a custom parameter to the data request, use the **addParams** method of **Query** class. Assign the **Query** object with additional parameters to the grid [query](#) property.

INDEX.JS

```
var data = new ej.data.DataManager({
  url: 'https://services.syncfusion.com/js/production/api/Orders',
  adaptor: new ej.data.ODataAdaptor()
});
var grid = new ej.grids.Grid({
  dataSource: data,
  query: new ej.data.Query().addParams('ej2grid', 'true'),
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign:
'Right', width: 120, type: 'number' },
    { field: 'CustomerID', width: 140, headerText: 'Customer
ID', type: 'string' },
    { field: 'Freight', headerText: 'Freight', textAlign:
'Right', width: 120, format: 'C' },
    { field: 'OrderDate', headerText: 'Order Date', width: 140,
format: 'yMd' }
  ]
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}

```

```

    </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The parameters added using the [query](#) property will be sent along with the data request for every grid action.

Handling HTTP error

During server interaction from the grid, some server-side exceptions may occur, and you can acquire those error messages or exception details

in client-side using the [actionFailure](#) event.

The argument passed to the [actionFailure](#) event contains the error details returned from the server.

The [actionFailure](#) event will be triggered not only for the server errors, but also when there is an exception while processing the grid actions.

Binding with ajax

You can use Grid [dataSource](#) property to bind the datasource to Grid from external ajax request. In the below code we have fetched the datasource from the server with the help of ajax request and provided that to [dataSource](#) property by using **onSuccess** event of the ajax.

```

`ts
import { Grid, Page } from '@syncfusion/ej2-grids';
import { Ajax } from '@syncfusion/ej2-base';
Grid.Inject(Page);
let grid: Grid = new Grid({
allowPaging: true,
columns: [
{ field: 'OrderID', headerText: 'Order ID', textAlign: 'Right', width: 120 },
{ field: 'CustomerID', headerText: 'Customer ID', textAlign: 'Right', width: 120 },
{ field: 'EmployeeID', headerText: 'Employee ID', textAlign: 'Right', width: 120 },
{ field: 'ShipCountry', headerText: 'Ship Country', textAlign: 'Right', width: 120 }
]
});
grid.appendTo('#Grid');
let button = document.getElementById('btn');
button.addEventListener("click", function(e){
let ajax = new Ajax("https://ej2services.syncfusion.com/production/web-services/api/Orders", "GET");
ajax.send();
ajax.onSuccess = function (data: string) {

```

```
grid.dataSource = JSON.parse(data);
};
});
`
```

* If you bind the dataSource from this way, then it acts like a local dataSource. So you cannot perform any server side crud actions.

[See Also](#)

Local data in `##Platform_Name##` Grid control

To bind local data to the grid, you can assign a JavaScript object array to the [dataSource](#) property. The local data source can also be provided as an instance of the [DataManager](#).

INDEX.JS

```
var gridData = data.slice(0, 7);
var grid = new ej.grids.Grid({
  dataSource: gridData,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign:
'Right', width: 120, type: 'number' },
    { field: 'CustomerID', width: 140, headerText: 'Customer
ID', type: 'string' },
    { field: 'Freight', headerText: 'Freight', textAlign:
'Right', width: 120, format: 'C' },
    { field: 'OrderDate', headerText: 'Order Date', width: 140,
format: 'yMd' }
  ]
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

By default, [DataManager](#) uses **JsonAdaptor** for local data-binding.

Refresh the data source

You can add/delete the data source records through an external button. To reflect the data source changes in the grid, invoke the [refresh](#) method.

To refresh the data source:

Step 1:

Add/delete the data source record by using the following code.

```
`ts
grid.dataSource.unshift(data); // add a new record.
grid.dataSource.splice(selectedRow, 1); // delete a record.
`
```

Step 2:

Refresh the grid after the data source change by using the [refresh](#) method.

```
`ts
grid.refresh(); // refresh the Grid.
`
```

INDEX.JS

```
var grid = new ej.grids.Grid({
  dataSource: data,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
    { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
    { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C2' },
    { field: 'OrderDate', headerText: 'Order Date', textAlign: 'Right',
width: 140, format: 'yMd' }
  ],
  height: 280
});
grid.appendTo('#Grid');
var add = new ej.buttons.Button({}, '#add');
var dele = new ej.buttons.Button({}, '#delete');
document.getElementById('add').onclick = function() {
  var data = { OrderID: 10247, CustomerID: "ASDFG", Freight: 40.4,
OrderDate: new Date(8367642e5) };
  (grid.dataSource).unshift(data); // add new record.
  grid.refresh(); // refresh the Grid.
};
document.getElementById('delete').onclick = function() {
  if (grid.getSelectedRowIndex().length) {
    var selectedRow = grid.getSelectedRowIndex()[0];
    (grid.dataSource).splice(selectedRow, 1); // delete the selected
record.
  } else {
    alert("No records selected for delete operation");
  }
  grid.refresh(); // refresh the Grid.
};
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <button class="e-flat" id="add">Add</button>
    <button class="e-flat" id="delete">Delete</button>
    <div id="Grid"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Remote data in ##Platform_Name## Grid control

To bind remote data to grid component, assign service data as an instance of [DataManager](#) to the [dataSource](#) property. To interact with remote data source, provide the endpoint url.

INDEX.JS

```
var data = new ej.data.DataManager({
  url: 'https://services.syncfusion.com/js/production/api/Orders',
  adaptor: new ej.data.ODataAdaptor()
});
var grid = new ej.grids.Grid({
  dataSource: data,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign:
'Right', width: 120, type: 'number' },
    { field: 'CustomerID', width: 140, headerText: 'Customer
ID', type: 'string' },
    { field: 'Freight', headerText: 'Freight', textAlign:
'Right', width: 120, format: 'C' },
    { field: 'OrderDate', headerText: 'Order Date', width: 140,
format: 'yMd' }
  ]
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

By default, [DataManager](#) uses **ODataAdaptor** for remote data-binding.

OData adaptor - Binding OData service

[OData](#) is a standardized protocol for creating and consuming data. You can retrieve data from OData service using the DataManager. Refer to the following code example for remote Data binding using OData service.

INDEX.JS

```

var data = new ej.data.DataManager({
    url: 'https://services.syncfusion.com/js/production/api/Orders',
    adaptor: new ej.data.ODataAdaptor(),
    crossDomain: true
});
var grid = new ej.grids.Grid({

```



```

        dataSource: data,
        columns: [
            { field: 'OrderID', headerText: 'Order ID', textAlign:
'Right', width: 120, type: 'number' },
            { field: 'CustomerID', width: 140, headerText: 'Customer
ID', type: 'string' },
            { field: 'Freight', headerText: 'Freight', textAlign:
'Right', width: 120, format: 'C' },
            { field: 'OrderDate', headerText: 'Order Date', width: 140,
format: 'yMd' }
        ]
    });
    grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <style>
        .e-row[aria-selected="true"] .e-customizedExpandcell {
            background-color: #e0e0e0;
        }
        .e-grid.e-gridhover tr[role='row']:hover {
            background-color: #eee;
        }
    </style>

```

```

        .e-expand::before {
            content: '\e5b8';
        }
        .empImage {
            margin: 6px 16px;
            float: left;
            width: 50px;
            height: 50px;
        }
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

OData v4 adaptor - Binding OData v4 service

The ODataV4 is an improved version of OData protocols, and the [DataManager](#) can also retrieve and consume OData v4 services. For more details on OData v4 services, refer to the [odata documentation](#). To bind OData v4 service, use the **ODataV4Adaptor**.

INDEX.JS

```

var data = new ej.data.DataManager({
    url:
    'https://services.odata.org/V4/Northwind/Northwind.svc/Orders/?$top=7',
    adaptor: new ej.data.ODataV4Adaptor()
});
var grid = new ej.grids.Grid({
    dataSource: data,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign:
        'Right', width: 120, type: 'number' },
        { field: 'CustomerID', width: 140, headerText: 'Customer
        ID', type: 'string' },
        { field: 'Freight', headerText: 'Freight', textAlign:
        'Right', width: 120, format: 'C' },
        { field: 'OrderDate', headerText: 'Order Date', width: 140,
        format: 'yMd' }
    ]
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
      background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
      background-color: #eee;
    }
    .e-expand::before {
      content: '\e5b8';
    }
    .empImage {
      margin: 6px 16px;
      float: left;
      width: 50px;
      height: 50px;
    }
  </style>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>

```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Web API adaptor

You can use **WebApiAdaptor** to bind grid with Web API created using OData endpoint.

```
`ts
```

```

import { Grid } from '@syncfusion/ej2-grids';
import { DataManager, WebApiAdaptor } from '@syncfusion/ej2-data';

let data: DataManager = new DataManager({
url: '/api/OrderAPI',
adaptor: new WebApiAdaptor
});

let grid: Grid = new Grid({
dataSource: data,
columns: [
{ field: 'OrderID', headerText: 'Order ID', textAlign: 'Right', width: 120, type: 'number' },
{ field: 'CustomerID', width: 140, headerText: 'Customer ID', type: 'string' },
{ field: 'Freight', headerText: 'Freight', textAlign: 'Right', width: 120, format: 'C' },
{ field: 'OrderDate', headerText: 'Order Date', width: 140, format: 'yMd' }
]
});

grid.appendTo('#Grid');
`

```

The response object should contain properties, **Items** and **Count**, whose values are a collection of entities and total count of the entities, respectively.

The sample response object should look like this:

```

、
{
Items: [{..}, {..}, {..}, ...],
Count: 830
}
、

```

Remote save adaptor

You may need to perform all Grid Actions in client-side except the CRUD operations, that should be interacted with server-side to persist data. It can be achieved in Grid by using **RemoteSaveAdaptor**.

Datasource must be set to **json** property and set **RemoteSaveAdaptor** to the **adaptor** property. CRUD operations can be mapped to server-side using **updateUrl**, **insertUrl**, **removeUrl**, **batchUrl**, **crudUrl** properties.

You can use the following code example to use **RemoteSaveAdaptor** in Grid.

```

`ts
import { Grid, Edit, Toolbar } from '@syncfusion/ej2-grids';
import { DataManager, RemoteSaveAdaptor } from '@syncfusion/ej2-data';
import { data } from './datasource.ts';

Grid.Inject(Edit, Toolbar);

let dataSource: DataManager = new DataManager({
json: data,
adaptor: new RemoteSaveAdaptor,
insertUrl: '/Home/Insert',
updateUrl: '/Home/Update',
removeUrl: '/Home/Delete'
});

let grid: Grid = new Grid({
dataSource: dataSource,
toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
editSettings: { allowEditing: true, allowAdding: true, allowDeleting: true },
columns: [
{ field: 'OrderID', headerText: 'Order ID', isPrimaryKey: true, textAlign: 'Right', width: 120, type: 'number'
},
{ field: 'CustomerID', width: 140, headerText: 'Customer ID', type: 'string' },
{ field: 'Freight', headerText: 'Freight', textAlign: 'Right', width: 120, format: 'C' },
{ field: 'OrderDate', headerText: 'Order Date', width: 140, format: 'yMd' }
]
}
)

```

```

]
});
grid.appendTo('#Grid');
`

```

The following code example describes the CRUD operations handled at server-side.

```

`
public ActionResult Update(OrdersDetails value)
{
    ...
    return Json(value);
}
public ActionResult Insert(OrdersDetails value)
{
    ...
    return Json(value);
}
public ActionResult Delete(int key)
{
    ...
    return Json(data);
}
`

```

Custom adaptor

You can create your own adaptor by extending the built-in adaptors. The following demonstrates custom adaptor approach and how to add a serial number for the records by overriding the built-in response processing using the **processResponse** method of the **ODataAdaptor**.

INDEX.JS

```

class SerialNoAdaptor extends ej.data.ODataV4Adaptor {
    processResponse() {
        var i= 0;
        // calling base class processResponse function
        var original = super.processResponse.apply(this, arguments);
        // adding serial number
        original.result.forEach(function(item){item['Sno'] = ++i});
        return { result: original.result, count: original.count };
    }
}
var data = new ej.data.DataManager({
    url: 'https://services.syncfusion.com/js/production/api/Orders',

```

```

    adaptor: new SerialNoAdaptor()
});
var grid = new ej.grids.Grid({
    dataSource: data,
    columns: [
        { field: 'Sno', width: 120, headerText: 'SNO', textAlign: 'Right' },
        { field: 'CustomerID', width: 140, headerText: 'Customer Name',
type: 'string' },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C2' },
        { field: 'OrderDate', headerText: 'Order Date', width: 140, format:
'yMd' }
    ]
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <style>
        .e-row[aria-selected="true"] .e-customizedExpandcell {
            background-color: #e0e0e0;
        }
        .e-grid.e-gridhover tr[role='row']:hover {

```

```

        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Offline mode

On remote data binding, all grid actions such as paging, sorting, editing, grouping, filtering, etc, will be processed on server-side. To avoid postback for every action, set the grid to load all data on initialization and make the actions process in client-side. To enable this behavior, use the **offline** property of [DataManager](#).

INDEX.JS

```

var data = new ej.data.DataManager({
    url: 'https://services.syncfusion.com/js/production/api/Orders',
    adaptor: new ej.data.ODataAdaptor(),
    offline: true
});
ej.grids.Grid.Inject(ej.grids.Page, ej.grids.Sort, ej.grids.Group);
var grid = new ej.grids.Grid({
    dataSource: data,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign:
'Right', width: 120, type: 'number' },
        { field: 'CustomerID', width: 140, headerText: 'Customer
ID', type: 'string' },
        { field: 'Freight', headerText: 'Freight', textAlign:
'Right', width: 120, format: 'C' },
    ]

```



```

        { field: 'OrderDate', headerText: 'Order Date', width: 140,
format: 'yMd' }
    ],
    allowPaging: true,
    pageSettings: { pageSize: 7 },
    allowSorting: true,
    allowGrouping: true
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
      background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
      background-color: #eee;
    }
    .e-expand::before {
      content: '\e5b8';
    }
    .empImage {

```

```

        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Immutable mode in ##Platform_Name## Grid control

To enable this feature, you have to set the [enableImmutableMode](#) property as **true**.

* This feature uses the primary key value for data comparison. So, you need to provide the [isPrimaryKey](#) column.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Page);
var immutableStart;
var normalStart;
var primaryKey = 0;
var immutableInit = true;
var init = true;
var immutableGrid = new ej.grids.Grid({
    dataSource: data,
    allowPaging: true,
    pageSettings: { pageSize: 50 },
    enableImmutableMode: true,
    beforeDataBound: function() {
        immutableStart = new Date().getTime();
    },
    dataBound: function() {
        var val = immutableInit ? '' : new Date().getTime() -
immutableStart;
        document.getElementById('immutableDelete').innerHTML = 'Immutable
rendering Time: ' + "<b>" + val + "</b>" + '<b>ms</b>';
        immutableStart = 0; immutableInit = false;
    },
    columns: [

```

```

        { field: 'OrderID', headerText: 'Order ID', isPrimaryKey: true,
textAlign: 'Right', width: 120 },
        { field: 'ProductName', headerText: 'Product Name', width: 160 },
        { field: 'ProductID', headerText: 'Product ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
        { field: 'CustomerName', headerText: 'Customer Name', width: 160 }
    ],
    height: 240
});
immutableGrid.appendTo('#immutable');
document.getElementById('delete').addEventListener('click', function(e) {
    immutableGrid.dataSource.splice(0, 5);
    normalGrid.setProperties({ dataSource: immutableGrid.dataSource });
    immutableGrid.setProperties({ dataSource: immutableGrid.dataSource });
});
document.getElementById('addtop').addEventListener('click', function(e) {
    var addedRecords = [
        { 'OrderID': ++primaryKey, 'ProductName': 'Chai', 'ProductID':
'Sasquatch Ale', 'CustomerID': 'QUEDE', 'CustomerName': 'Yoshi Tannamuri' },
        { 'OrderID': ++primaryKey, 'ProductName': 'Georg Pipp's',
'ProductID': 'Valkoinen suklaa', 'CustomerID': 'RATTC', 'CustomerName':
'Martin Sommer' },
        { 'OrderID': ++primaryKey, 'ProductName': 'Yoshi Tannamuri',
'ProductID': 'Gula Malacca', 'CustomerID': 'COMMI', 'CustomerName': 'Ann
Devon' },
        { 'OrderID': ++primaryKey, 'ProductName': 'Palle Ibsen',
'ProductID': 'Rogede sild', 'CustomerID': 'RATTC', 'CustomerName': 'Paula
Wilson' },
        { 'OrderID': ++primaryKey, 'ProductName': 'Francisco Chang',
'ProductID': 'Mascarpone Fabioli', 'CustomerID': 'ROMEY', 'CustomerName':
'Jose Pavarotti' }
    ]
    var aData = addedRecords.concat(immutableGrid.dataSource);
    normalGrid.setProperties({ dataSource: aData });
    immutableGrid.setProperties({ dataSource: aData });
});
document.getElementById('addbottom').addEventListener('click', function(e) {
    var addedRecords = [
        { 'OrderID': ++primaryKey, 'ProductName': 'Chai', 'ProductID':
'Sasquatch Ale', 'CustomerID': 'QUEDE', 'CustomerName': 'Yoshi Tannamuri' },
        { 'OrderID': ++primaryKey, 'ProductName': 'Georg Pipp's',
'ProductID': 'Valkoinen suklaa', 'CustomerID': 'RATTC', 'CustomerName':
'Martin Sommer' },
        { 'OrderID': ++primaryKey, 'ProductName': 'Yoshi Tannamuri',
'ProductID': 'Gula Malacca', 'CustomerID': 'COMMI', 'CustomerName': 'Ann
Devon' },
        { 'OrderID': ++primaryKey, 'ProductName': 'Palle Ibsen',
'ProductID': 'Rogede sild', 'CustomerID': 'RATTC', 'CustomerName': 'Paula
Wilson' },
        { 'OrderID': ++primaryKey, 'ProductName': 'Francisco Chang',
'ProductID': 'Mascarpone Fabioli', 'CustomerID': 'ROMEY', 'CustomerName':
'Jose Pavarotti' }
    ]
    var aData = immutableGrid.dataSource.concat(addedRecords);
    normalGrid.setProperties({ dataSource: aData });
    immutableGrid.setProperties({ dataSource: aData });
});

```

```

});
document.getElementById('reverse').addEventListener('click', function(e) {
    var aData = immutableGrid.dataSource.reverse();
    normalGrid.setProperties({ dataSource: aData });
    immutableGrid.setProperties({ dataSource: aData });
});
document.getElementById('paging').addEventListener('click', function(e) {
    var totalPages = immutableGrid.dataSource.length /
immutableGrid.pageSettings.pageSize;
    var page = Math.floor(Math.random() * totalPages) + 1;
    normalGrid.setProperties({ pageSettings: { currentPage: page } });
    immutableGrid.setProperties({ pageSettings: { currentPage: page } });
});
var normalGrid = new ej.grids.Grid({
    dataSource: data,
    allowPaging: true,
    pageSettings: { pageSize: 50 },
    beforeDataBound: function() {
        normalStart = new Date().getTime();
    },
    dataBound: function() {
        var val = init ? '' : new Date().getTime() - normalStart;
        document.getElementById('normalDelete').innerHTML = 'Normal
rendering Time: ' + "<b>" + val + "</b>" + '<b>ms</b>';
        normalStart = 0; init = false;
    },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', isPrimaryKey: true,
textAlign: 'Right', width: 120 },
        { field: 'ProductName', headerText: 'Product Name', width: 160 },
        { field: 'ProductID', headerText: 'Product ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
        { field: 'CustomerName', headerText: 'Customer Name', width: 160 }
    ],
    height: 240
});
normalGrid.appendTo('#normal');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .immutablegrid,
    .normalgrid {
        pointer-events: none;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <script id="template" type="text/x-template">
            <div class="image">
                
            </div>
        </script>
        <table>
            <tbody>
                <tr>
                    <td>
                        <span id="immutableDelete"></span>
                    </td>
                </tr>
                <tr>
                    <td>
                        <span id="normalDelete"></span>
                    </td>
                </tr>
                <tr>
                    <td>
                        <div>
                            <button id="addtop" class="e-control e-btn e-lib
e-info">Add 5 rows at top</button>
                            <button id="addbottom" class="e-control e-btn e-
lib e-info">Add 5 rows at bottom</button>
                            <button id="delete" class="e-control e-btn e-lib
e-info">Delete 5 rows</button>
                        </div>
                    </td>
                </tr>
            </tbody>
        </table>
    </div>

```

```

        <button id="reverse" class="e-control e-btn e-lib e-info">Sort Order ID</button>
        <button id="paging" class="e-control e-btn e-lib e-info">Paging</button>
    </div>
</td>
</tr>
<tr>
    <td>
        <span><b>Immutable Grid</b></span>
        <div id="immutable" class="immutablegrid"></div>
    </td>
</tr>
<tr>
    <td>
        <span><b>Normal Grid</b></span>
        <div id="normal" class="normalgrid"></div>
    </td>
</tr>
</tbody>
</table>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Limitations

The following features are not supported in the immutable mode:

- Frozen rows and columns
- Row Template
- Detail Template
- Hierarchy Grid
- Column reorder
- Virtual scroll
- Infinite scroll
- Grouping

Performance tips for ##Platform_Name## Grid control

This article is a comprehensive guide on improving the loading performance of the ##PlatformName## Grid, especially when dealing with large datasets along with large number of columns. It provides valuable insights into the steps that need to be followed to bind a large data source without experiencing any performance degradations. By offering detailed explanations and actionable tips, this resource aims to empower readers with the knowledge and best practices necessary to optimize the performance of the ##PlatformName## Grid during data binding, ensuring a smooth and efficient user experience.

How to improve loading performance by binding large dataset

As you all know, a grid is made up of rows and columns. For instance, when you bind 10 rows and 10 columns, it means 100 elements will be rendered in the DOM (Document Object Model). So, it is recommended to render only a limited number of rows and columns to guarantee the best loading performance for the control.

Optimizing performance with paging

To boost the performance efficiency of your application, especially when dealing with large datasets, it is advised to implement paging. [Paging](#) allows you to display grid data in segmented pages, facilitating easier navigation through substantial datasets. This feature proves particularly beneficial in enhancing the overall performance of your application. For more information on implementing paging, you can refer to the [documentation](#) section dedicated to this feature.

Optimizing performance with row virtualization or infinite scrolling

To enhance your application's efficiency, especially when dealing with substantial datasets, it is recommended to either using [virtualization](#) or [infinite scrolling](#). Implementing these techniques can significantly reduce the load on your application and elevate its overall performance.

1. **Virtualization:** The Virtual scrolling feature in the ##Platform_Name## Grid enables the efficient handling and display of large volumes of data without compromising performance. This approach optimizes the rendering process by loading only the visible rows within the Grid viewport, rather than rendering the entire dataset simultaneously. For more information on implementing row virtualization , you can refer to the [documentation](#) section dedicated to this feature.
2. **Infinite scrolling:** The Infinite Scrolling feature in the ##Platform_Name## Grid is a powerful tool for seamlessly handling extensive data sets without compromising grid performance. It operates on a "load-on-demand" concept, ensuring that data is fetched only when needed. In the default infinite scrolling mode, a new block of data is loaded each time the scrollbar reaches the end of the vertical scroller. For more information on implementing infinite scrolling , you can refer to the [documentation](#) section dedicated to this feature.

Optimizing performance with column virtualization in large no of columns

[Column virtualization](#) feature in the ##Platform_Name## Grid that allows you to optimize the rendering of columns by displaying only the columns that are currently within the viewport. It allows horizontal scrolling to view additional columns. This feature is particularly useful when dealing with grids that have a large number of columns, as it helps to improve the performance and reduce the initial loading time.

It is possible to enable both row and column virtualization. This feature allows for efficient handling of large datasets by dynamically loading only the visible rows and columns, optimizing performance and enhancing the overall responsiveness of the grid. For more information on implementing column virtualization , you can refer to the [documentation](#) section dedicated to this feature.

How to overcome browser height limitation in virtual scrolling

[Documentation link](#)

How to improve loading performance by binding large data by showing custom text or element

When integrating image or template elements into a column, it's recommended to utilize the [Column Template](#) feature rather than customizing the data through [rowDataBound](#) or [queryCellInfo](#) events. These events are triggered for each row and cell rendering, introducing delays in the control's rendering process. Moreover, rendering custom elements using these events may result in the persistence of

rendered elements, potentially causing longer rendering times over time. By opting for the column template feature, you can efficiently meet this requirement without experiencing rendering delays and ensure a more streamlined rendering process.

How to improve loading performance by referring individual script and CSS

To improve the performance of Syncfusion Grid control during the initial render as well as certain actions, suggested you to download the specific control scripts using CRG (Custom Resource Generator) to speed up the project. By default, the ej2.min.js script file contains all the Syncfusion control scripts. So, it will take some time to load the scripts to the project. Using [CRG](#), you can select the controls which you want to use, and the modules for those controls, then you can download the scripts and CSS for the selected controls and use them as per your need.

[CRG website link](#)

So to improve the performance of grid during the initial rendering, suggested you to refer individual script and CSS.

How to update cell values without frequent server calls

Efficiently update cell values without the need for frequent server calls, especially beneficial for live update scenarios. Even when the data is initially bound from the server, performing edit operations can be done without triggering a database refresh. Utilize the [setCellValue](#) method to update the Grid without affecting the database and only refresh the UI.

How to optimize server-side data operations with adaptors

The ##Platform_Name## Grid provides support for various adaptors (OData, ODataV4, WebAPI, URL, etc.) to facilitate server-side data operations and CRUD functionalities. By leveraging these adaptors along with the `DataManager` control, you can seamlessly bind remote data sources to the grid and execute actions. During data operations like filtering, sorting, and paging, the corresponding action queries are generated as per the adaptor's requirements. It is crucial to handle these actions on the application end and return the processed data back to the grid. Refer to the documentation for comprehensive details. It's worth noting that for efficient data processing, the suggested order for returning processed data to the grid is as follows

- Filtering
- Sorting
- Aggregates
- Paging
- Grouping

How to avoid MaxJsonLength error while passing large amount of records

The ##Platform_Name## Grid control is client-server based. So, we send the data as JSON object between client and server. The reported issue occurs due to the serialization of the large-sized JSON object. We need to increase the maximum length for serializing the large-sized JSON object. You have to alter the [MaxJsonLength](#) property on your web.config file or in the place of deserialization.

Solution: 1

```
`csharp
<configuration>
<system.web.extensions>
```



```

<scripting>
<webServices>
<jsonSerialization maxJsonLength="25000000"/>
</webServices>
</scripting>
</system.web.extensions>
</configuration>
`

```

Solution : 2

```

`csharp
var serializer = new JavaScriptSerializer { MaxJsonLength = Int32.MaxValue };
`

```

Microsoft excel limitation while exporting millions of records to excel file format

By default, Microsoft Excel supports only 1,048,576 records in an excel sheet. Hence it is not possible to export millions of records to excel. You can refer the [documentation](#) link for more details on Microsoft excel specifications and limits. So suggest to export the data in CSV (Comma-Separated Values) or other formats that can handle large datasets more efficiently than Excel.

Columns

Columns in ##Platform_Name## Grid control

The column definitions are used as the [dataSource](#) schema in the Grid. This plays a vital role in rendering column values in the required format. The grid operations such as sorting, filtering and grouping etc. are performed based on column definitions. The [field](#) property of the [columns](#) is necessary to map the data source values in Grid columns.

1. If the column [field](#) is not specified in the dataSource, the column values will be empty.
2. If the [field](#) name contains “dot” operator, it is considered as complex binding.

Column types

Column type can be specified using the [columns.type](#) property. It specifies the type of data the column binds.

If the [format](#) is defined for a column, the column uses [type](#) to select the appropriate format option ([number](#) or [date](#)).

Grid column supports the following types:

- string
- number
- boolean
- date
- datetime

If the [type](#) is not defined, it will be determined from the first record of the [dataSource](#).

Incase if the first record of the [dataSource](#) is null/blank value for a column then it is necessary to define the [type](#) for that column.

ValueAccessor

The [valueAccessor](#) is used to access/manipulate the value of display data. You can achieve custom value formatting by using the [valueAccessor](#).

INDEX.JS

```
var grid = new ej.grids.Grid({
  dataSource: data,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
    { field: 'Freight', headerText: 'Freight', width: 100,
valueAccessor: currencyFormatter },
    { field: 'ShipName', headerText: 'Ship Name', width: 180,
valueAccessor: valueAccess },
  ],
  height: 315
});
grid.appendTo('#Grid');
function currencyFormatter(field, data, column) {
  return '€' + data['Freight'];
}
function valueAccess(field, data, column) {
  return data[field] + '-' + data['ShipRegion'];
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Display array type columns

You can bind an array of objects in a column by using the [valueAccessor](#) property. In this example, the name field has an array of two objects, FirstName and LastName. These two objects are joined and bound to a column using the [valueAccessor](#).

INDEX.JS

```

var grid = new ej.grids.Grid({
    dataSource: stringData,
    columns: [
        { field: 'EmployeeID', headerText: 'Employee ID', textAlign:
'Right', width: 100 },
        { field: 'Name', headerText: 'Full Name', width: 120, valueAccessor:
valueAccess },
        { field: 'Title', headerText: 'Title', width: 150 }
    ],
    height: 315
});
grid.appendTo('#Grid');
function valueAccess(field, data, column){
    return data[field].map(function (s) {
        return s.LastName || s.FirstName }).join(' ');
    }
}

```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Grid"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Expression column

You can achieve the expression column by using the [valueAccessor](#) property.

INDEX.JS

```
var grid = new ej.grids.Grid({
  dataSource: foodInformation,
  columns: [
    { field: 'FoodName', headerText: 'FoodName', width: 120 },
    { field: 'Protein', headerText: 'Protein', textAlign: 'Right',
width: 100 },
    { field: 'Fat', headerText: 'Fat', textAlign: 'Right', width: 100 },
    { field: 'Carbohydrate', headerText: 'Carbohydrate', textAlign:
'Right', width: 130 },
    { headerText: 'Calories In Take', valueAccessor: totalCalories,
textAlign: 'Right', width: 150 },
  ],
  height: 315
});
grid.appendTo('#Grid');
function totalCalories(field, data, column){
  return data.Protein * 4 + data.Fat * 9 + data.Carbohydrate * 4;
};
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Format

To format cell values based on specific culture, use the [columns.format](#) property. The grid uses [Internalization](#) library to format [number](#) and [date](#) values.

INDEX.JS

```

var grid = new ej.grids.Grid({
    dataSource: data,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign:
'Right', width: 120 },
        { field: 'CustomerID', width: 140, headerText: 'Customer ID'
},
        { field: 'Freight', headerText: 'Freight', textAlign:
'Right', width: 120, format: 'C' },
        { field: 'OrderDate', headerText: 'Order Date', textAlign:
'Right', width: 140, format: 'yMd' }
    ],
    height: 315
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">

```

```

<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');

```

```

if(ele) {
    ele.style.visibility = "visible";
}

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

By default, the [number](#) and [date](#) values are formatted in **en-US** locale. You can localize the currency and date in different locale as explained [here](#)

Number formatting

The number or integer values can be formatted using the below format strings.

Format | Description | Remarks

{ type:'date', format:'dd/MM/yyyy' } | 04/07/1996

{ type:'date', format:'dd.MM.yyyy' } | 04.07.1996

{ type:'date', skeleton:'short' } | 7/4/96

{ type: 'dateTime', format: 'dd/MM/yyyy hh:mm a' } | 04/07/1996 12:00 AM

{ type: 'dateTime', format: 'MM/dd/yyyy hh:mm:ss a' } | 07/04/1996 12:00:00 AM

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Page);
var grid = new ej.grids.Grid({
    dataSource: data,
    allowPaging: true,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'Freight', headerText: 'Freight', format:'C2', width: 120
},
        { field: 'OrderDate',format: {type:'date', format:'dd/MM/yyyy'},
headerText:'Order Date', width:150},
        { field: 'OrderDate', format:{ type: 'dateTime', format: 'dd/MM/yyyy
hh:mm a' }, headerText: 'Ship Date', width: 180 }
    ],
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Render boolean values as checkbox

To render boolean values as checkbox in columns, you need to set [displayAsCheckBox](#) property as **true**.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Edit);
var grid = new ej.grids.Grid({
    dataSource: data,
    editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100, isPrimaryKey: true },

```

```

        { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C2' },
        { field: 'ShipCountry', headerText: 'Ship Country', width: 150 },
        { field: 'Verified', headerText: 'Verified', width: 150,
displayAsCheckBox: true }
    ],
    height: 315
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
      background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
      background-color: #eee;
    }
    .e-expand::before {
      content: '\e5b8';
    }
  </style>

```

```

        .empImage {
            margin: 6px 16px;
            float: left;
            width: 50px;
            height: 50px;
        }
    </style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Visibility

You can hide any particular column in Grid before rendering by defining [visible](#) property as false. In the below sample **ShipCity** column is defined as visible false.

INDEX.JS

```

var grid = new ej.grids.Grid({
    dataSource: data,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
        { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
        { field: 'ShipCity', width: 140, headerText: 'Ship City', visible:
false },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C' },
        { field: 'OrderDate', headerText: 'Order Date', width: 140, format:
'yMd', textAlign: 'Right' }
    ],
    height: 315
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>

```

```

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
.e-row[aria-selected="true"] .e-customizedExpandcell {
    background-color: #e0e0e0;
}
.e-grid.e-gridhover tr[role='row']:hover {
    background-color: #eee;
}
.e-expand::before {
    content: '\e5b8';
}
.empImage {
    margin: 6px 16px;
    float: left;
    width: 50px;
    height: 50px;
}
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

<div id="container">

```

```

        <div id="Grid"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Lock columns

You can lock columns by using [column.lockColumn](#) property. The locked columns will be moved to the first position. Also you can't reorder its position.

In the below example, **Ship City** column is locked and its reordering functionality is disabled.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Reorder);
var grid = new ej.grids.Grid({
    dataSource: data,
    allowReordering: true,
    allowSelection: false,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
        { field: 'ShipCity', headerText: 'Ship City',width: 100, lockColumn:
true, customAttributes: { class: 'customcss' } },
        { field: 'ShipName', headerText: 'Ship Name', width: 100 },
        { field: 'ShipPostalCode', headerText: 'Ship Postal code', width:
100 },
        { field: 'ShipRegion', headerText: 'Ship Region', width: 100 }
    ],
    height: 315
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="lock.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
#reorderMultipleCols {
    text-transform: none;
}
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <script id="rowtemplate" type="text/x-template">
        <tr>
            <td class="photo">
                
            </td>
            <td class="details">
                <table class="CardTable" cellpadding="3" cellspacing="2">
                    <colgroup>
                        <col width="50%">
                        <col width="50%">
                    </colgroup>
                    <tbody>
                        <tr>
                            <td class="CardHeader">First Name </td>
                            <td>{FirstName} </td>
                        </tr>
                        <tr>
                            <td class="CardHeader">Last Name</td>
                            <td>{LastName} </td>
                        </tr>
                        <tr>
                            <td class="CardHeader">Title
                            </td>
                            <td>{Title}
                            </td>
                        </tr>
                    </tbody>
                </table>
            </td>
        </tr>
    </script>

```

```

        <tr>
            <td class="CardHeader">Country
            </td>
            <td>${Country}
            </td>
        </tr>
    </tbody>
</table>
</td>
</tr>
</script>

<div id="container">
    <div id="Grid"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Controlling Grid actions

You can enable or disable grid action for a particular column by setting the [allowFiltering](#), [allowGrouping](#), [allowSorting](#), [allowReordering](#), and [allowEditing](#) properties.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Group, ej.grids.Filter, ej.grids.Sort,
ej.grids.Edit, ej.grids.Reorder);
var grid = new ej.grids.Grid({
    dataSource: data,
    allowFiltering: true,
    allowGrouping: true,
    allowSorting: true,
    allowReordering: true,
    editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true, mode: 'Normal' },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', allowGrouping: false,
textAlign: 'Right', width: 100 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
        { field: 'ShipCity', headerText: 'Ship City', allowFiltering: false,
allowReordering: false, allowEditing: false, width: 100 },
        { field: 'ShipName', headerText: 'Ship Name', allowSorting: false,
width: 100 }
    ],
    height: 220
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Grid"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```


Show or hide columns by external button

You can show or hide grid columns dynamically using external buttons by invoking the [showColumns](#) or [hideColumns](#) method.

INDEX.JS

```
var grid = new ej.grids.Grid({
  dataSource: data,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
    { field: 'ShipCity', headerText: 'Ship City', width: 100 },
    { field: 'ShipName', headerText: 'Ship Name', width: 100 }
  ],
  height: 280
});
grid.appendTo('#Grid');
var show = new ej.buttons.Button({ cssClass: 'e-flat' }, '#show');
var hide = new ej.buttons.Button({ cssClass: 'e-flat' }, '#hide');
document.getElementById('show').onclick = function() {
  grid.showColumns(['Customer ID', 'Ship Name']); //show by HeaderText
};
document.getElementById('hide').onclick = function() {
  grid.hideColumns(['Customer ID', 'Ship Name']); //hide by HeaderText
};
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="show">Show</button>
        <button id="hide">Hide</button>
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customize column styles

You can customize the appearance of the header and content of a particular column using the [customAttributes](#) property.

To customize the grid column, follow the given steps:

Step 1:

Create a CSS class with custom style to override the default style for rowcell and headercell.

```

.e-grid .e-rowcell.customcss{
background-color: #ecedee;
color: 'red';
font-family: 'Bell MT';
font-size: 20px;
}
.e-grid .e-headercell.customcss{
background-color: #2382c3;
color: white;
}

```

```
font-family: 'Bell MT';
font-size: 20px;
}
,
```

Step 2:

Add the custom CSS class to the specified column by using the [customAttributes](#) property.

```
`ts
{ field: 'ShipCity', headerText: 'Ship City', customAttributes: {class: 'customcss'}, width: 100 },
,
```

INDEX.JS

```
var grid = new ej.grids.Grid({
  dataSource: data,
  enableHover: false,
  allowSelection: false,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
    { field: 'ShipCity', headerText: 'Ship City', customAttributes: {
class: 'customcss' }, width: 100 },
    { field: 'ShipName', headerText: 'Ship Name', width: 100 }
  ],
  height: 315
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Display custom tooltip for columns

To display a custom ToolTip ([EJ2 Tooltip](#)), you can render the Grid control inside the Tooltip component and set the target as ".e-rowcell". The tooltip is displayed when hovering the grid cells.

Change the tooltip content for the grid cells by using the following code in the [beforeRender](#) event.

```

`ts
function beforeRender(args) {
// event triggered before render the tooltip on target element.
tooltip.content = args.target.closest("td").innerText;
}
`

```

INDEX.JS

```

var tooltip = new ej.popups.Tooltip({
    target: ".e-rowcell", // set the target element to show the tooltip on
    hovering it
    beforeRender: beforeRender
}, "#Tooltip");

```

```

var grid= new ej.grids.Grid({
    dataSource: data,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
        { field: 'ShipName', headerText: 'Ship Name', width: 140 },
        { field: 'ShipCity', headerText: 'Ship City', width: 100 },
    ],
    height: 315
});
grid.appendTo('#Grid');
function beforeRender(args) {
    // event triggered before render the tooltip on target element.
    tooltip.content = args.target.closest("td").innerText;
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```

```

<div id="container">
  <div id="Tooltip">
    <div id="Grid"></div>
  </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Align the text of Grid content and header

For aligning the text of Grid content and header part, kindly use [textAlign](#) and [headerTextAlign](#) properties.

Grid column supports the following alignments:

- Left
- Right
- Center
- Justify

INDEX.JS

```

var grid = new ej.grids.Grid({
  dataSource: data,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', type: 'number',
      textAlign: 'Right', headerTextAlign: 'Right', width: 120 },
    { field: 'CustomerID', headerText: 'Customer ID', type: 'string',
      textAlign: 'Left', headerTextAlign: 'Left', width: 90 },
    { field: 'OrderDate', headerText: 'Order Date', type: 'date',
      textAlign: 'Center', headerTextAlign: 'Center', format: 'yMd', width: 140 },
    { field: 'ShipCountry', headerText: 'Ship Country', type: 'string',
      textAlign: 'Justify', headerTextAlign: 'Justify', width: 120 }
  ],
  height: 315
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}

```

```

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

How to prevent checkbox in the blank row

By default, cells in the grid will be blank if the corresponding column values in the data source are null or undefined. The grid also has the option to prevent the rendering of checkboxes in such cases, even if the [displayAsCheckBox](#) property is set to true for that column, by using the [rowDataBound](#) event of the Grid.

In the following sample, the `rowDataBound` event of the Grid is used to set the innerHTML of the checkbox element to empty.

INDEX.JS

```

var grid= new ej.grids.Grid({
  dataSource: data,
  rowDataBound: rowDataBound,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100, isPrimaryKey: true },
    { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
    { field: 'Freight', headerText: 'Freight', width: 100, textAlign:
'Right' },
    { field: 'ShipName', headerText: 'Ship Name', width: 180 },
    { field: 'Verified', headerText: 'Verified', width: 150,
displayAsCheckBox: true },
  ]
});
grid.appendTo('#Grid');
function rowDataBound(args) {
  var count = 0;
  var keys = Object.keys(args.data);
  for (var i = 0; i < keys.length; i++) {
    if (args.data[keys[i]] == null || args.data[keys[i]] == '' ||
args.data[keys[i]] == undefined) {
      count++;
    }
  }
  if (count == keys.length) {
    for (var i = 0; i < grid.columns.length; i++) {
      if (grid.columns[i].displayAsCheckBox) {
        args.row.children[i].innerHTML = '';
      }
    }
  }
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">

```



```

<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Group Column by Format](#)
- [How to set complex column as Foreignkey column](#)
- [Complex Data Binding with list of Array Of Objects](#)
- [Compare columns effortlessly using column pinning feature in Grid](#)

- [How to edit the column in Grid using the uploader control](#)
- [How to add the aggregate option functionalities in the column menu's feature](#)
- [How to render tooltip with custom content on Grid columns](#)
- [How to display the multiplication table of a number accepted from the user](#)
- [How to drag and drop within grid and between grids](#)
- [How to format date values received in JSON data through ajax post](#)

Auto generated columns in ##Platform_Name## Grid control

The [columns](#) are automatically generated when [columns](#) declaration is empty or undefined while initializing the grid. All the columns in the [dataSource](#) are bound as grid columns.

INDEX.JS

```
var data = [
  { OrderID: 10248, CustomerID: 'VINET', EmployeeID: 5 },
  { OrderID: 10249, CustomerID: 'TOMSP', EmployeeID: 6 },
  { OrderID: 10250, CustomerID: 'HANAR', EmployeeID: 4 }];
var grid = new ej.grids.Grid ({
  dataSource: data
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <script id="rowtemplate" type="text/x-template">
    <tr>
      <td class="photo">
        
      </td>
      <td class="details">
        <table class="CardTable" cellpadding="3" cellspacing="2">
          <colgroup>
            <col width="50%">
            <col width="50%">
          </colgroup>
          <tbody>
            <tr>
              <td class="CardHeader">First Name </td>
              <td>{FirstName} </td>
            </tr>
            <tr>
              <td class="CardHeader">Last Name</td>
              <td>{LastName} </td>
            </tr>
            <tr>
              <td class="CardHeader">Title
              </td>
              <td>{Title}
              </td>
            </tr>
            <tr>
              <td class="CardHeader">Country
              </td>
              <td>{Country}
              </td>
            </tr>
          </tbody>
        </table>
      </td>
    </tr>
  </script>

  <div id="container">
    <div id="Grid"></div>
  </div>
</script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>

```

```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

When columns are auto-generated, the column [type](#) will be determined from the first record of the [dataSource](#).

Set [isPrimaryKey](#) for auto generated columns when editing is enabled

Primary key can be defined in the declaration of column object of the grid. When we didn't declare the columns, the grid will generate the columns automatically. For these auto generated columns, you can set [isPrimaryKey](#) column property as true by any one of the following two ways,

If Primary key "column index" is known then refer the following code example

```
`ts
import { Grid, Edit } from '@syncfusion/ej2-grids';
Grid.Inject(Edit);
let data: Object[] = [
{ OrderID: 10248, CustomerID: 'VINET', EmployeeID: 5 },
{ OrderID: 10249, CustomerID: 'TOMSP', EmployeeID: 6 },
{ OrderID: 10250, CustomerID: 'HANAR', EmployeeID: 4 }];
let grid: Grid = new Grid ({
dataSource: data,
editSettings: { allowEditing: true, allowAdding: true, allowDeleting: true }
});
grid.appendTo('#Grid');
grid.dataBound = () => {
var column: Column = grid.columns[0];
column.isPrimaryKey = 'true';
}
`
```

If Primary key "column fieldname" is known then refer the following code example

```
`ts
import { Grid, Edit } from '@syncfusion/ej2-grids';
Grid.Inject(Edit);
let data: Object[] = [
{ OrderID: 10248, CustomerID: 'VINET', EmployeeID: 5 },
{ OrderID: 10249, CustomerID: 'TOMSP', EmployeeID: 6 },
{ OrderID: 10250, CustomerID: 'HANAR', EmployeeID: 4 }];
```

```

let grid: Grid = new Grid ({
  dataSource: data,
  editSettings: { allowEditing: true, allowAdding: true, allowDeleting: true }
});

grid.appendTo('#Grid');

grid.dataBound = () => {
  var column: Column = grid.getColumnByField('OrderID');
  column.isPrimaryKey = 'true';
}

```

Set column options to auto generated columns

You can set column options such as [format](#), [width](#) to the auto generated columns by using [dataBound](#) event of the grid.

In the below example, [width](#) is set for **OrderID** column, **date** type is set for **OrderDate** column and **numeric** type is set for **Freight** column.

INDEX.JS

```

var data = [
  { OrderID: 10248, CustomerID: 'VINET', Freight: 32.3800,
    OrderDate: "1996-07-02T00:00:00.000Z" },
  { OrderID: 10249, CustomerID: 'TOMSP', Freight: 32.3800,
    OrderDate: "1996-07-19T00:00:00.000Z" },
  { OrderID: 10250, CustomerID: 'HANAR', Freight: 32.3800,
    OrderDate: "1996-07-22T00:00:00.000Z" }];
var grid = new ej.grids.Grid ({
  dataSource: data,
  dataBound: dataBound
});
grid.appendTo('#Grid');
function dataBound(args) {
  for (var i = 0; i < this.columns.length; i++) {
    this.columns[0].width = 120;
    if(this.columns[i].field === "OrderDate"){
      this.columns[i].type="date";
    }
    if (this.columns[i].type === "date") {
      this.columns[i].format = { type: "date", format:
"dd/MM/yyyy" };
    }
    this.columns[2].format = "P2";
  }
  this.refreshColumns();
}

```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
```

```

<title>EJ2 Grid</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <script id="rowtemplate" type="text/x-template">
    <tr>
      <td class="photo">
        
      </td>
      <td class="details">
        <table class="CardTable" cellpadding="3" cellspacing="2">
          <colgroup>
            <col width="50%">
            <col width="50%">
          </colgroup>
          <tbody>
            <tr>
              <td class="CardHeader">First Name </td>
              <td>{FirstName} </td>
            </tr>
          </tbody>
        </table>
      </td>
    </tr>
  </script>

```

```

        <td class="CardHeader">Last Name</td>
        <td>${LastName} </td>
    </tr>
    <tr>
        <td class="CardHeader">Title
        </td>
        <td>${Title}
        </td>
    </tr>
    <tr>
        <td class="CardHeader">Country
        </td>
        <td>${Country}
        </td>
    </tr>
</tbody>
</table>
</td>
</tr>
</script>

<div id="container">
    <div id="Grid"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Headers in ##Platform_Name## Grid control

Header text

By default, the header text of a column in Grid is displayed from the column's [field](#) value. However, you can easily override the default header title and provide a custom header text for the column using the [headerText](#) property.

To enable the `headerText` property, you simply need to define it in the [columns](#) property. The following example demonstrates how to enable header text for a Grid column.

INDEX.JS

```

var grid = new ej.grids.Grid({
    dataSource: data,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 140, },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
format: 'C', width: 120 },
        { field: 'OrderDate', headerText: 'Order Date', textAlign: 'Right',
format: 'yMd', width: 140 }
    ],

```

```

        height: 315
    });
    grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <div id="container">
        <div id="Grid"></div>
    </div>
    <script>
        var ele = document.getElementById('container');
        if (ele) {
            ele.style.visibility = "visible";
        }
    </script>
    <script src="index.js" type="text/javascript"></script>

```



```
</body>
</html>
```

* The **headerText** property is optional, and if it is not defined, then the corresponding column's field value is set as header text for that column.

* You can also use the [headerTemplate](#) property to apply custom HTML content to the header cell.

Header template

The [headerTemplate](#) property is used to customize the header element of a Grid column. With this property, you can render custom HTML elements or **##Platform_Name##** control to the header element. This feature allows you to add more functionality to the header, such as sorting or filtering.

In this demo, the custom element is rendered for both **CustomerID** and **OrderDate** column headers.

INDEX.JS

```
var grid = new ej.grids.Grid({
  dataSource: data,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right', width:
120, },
    {
      field: 'CustomerID', headerText: 'Customer ID', width: 140,
      headerTemplate: ` <div>
        <span class="e-icon-userlogin"></span>
        Customer ID
      </div>`,
    },
    { field: 'Freight', headerText: 'Freight', format: "C", width: 120,
      headerTemplate: ` <div id='Freight'></div>`, },
    {
      field: 'OrderDate', headerText: 'Order Date', textAlign: "Right",
      width: 200, format: 'yMd',
      headerTemplate: `<div><span id='OrderDate'></span>
        <label id='headerText' style='margin-left:8px'>Order Date</label>
      </div>`
    },
  ],
  height: 315
});
grid.appendTo('#Grid');
var dropdownData = ['Freight', 'Shipment', 'Cargo'];
var dropDownColumn = new ej.dropdowns.DropDownList({
  value: 'Freight',
  popupHeight: '200px',
  dataSource: dropdownData,
});
dropDownColumn.appendTo('#Freight');
var toggleButton = new ej.buttons.Switch({
  headerText: 'Order Date',
  change: onSwitchToggle,
});
toggleButton.appendTo('#OrderDate');
function onSwitchToggle(args) {
  headerText = args.checked ? 'Purchase Date' : 'Order Date';
```

```
document.getElementById('headerText').innerHTML = headerText;
}
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <style>
    @font-face {
      font-family: 'ej2-headertemplate';
      src:
        url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMj1vTFIAAAEoAAAAVmNtYXDS2c5qAAABjAAAAEBnbHlmQmN
FrQAAAdQAAANoaGVhZBRdbkIAAADQAAANmhoZWEIUAQAAAArAAAACRobXR4DAAAAAAAAAYAAAAA
MbG9jYQCOAbQAAAHMAAAACG1heHABHgENAAABCAAAACBuYW11ohGZJQAABTAAAAKpcG9zdA2o3w0
AAAFoAAAAQAABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAAwABAAAAQAATBXY9l8
PPPUACwQAAAAANillKkAAAAA2KWUqQAAAAAD9APzAAAACAACAAAAAAAAAAAAEAAAADAQEAEQAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wLpYAQAAAAAXAQAAAAAAAAABAAAAAAAAABAAAAQA
AAAAEAAAAAAAAAgAAAMAAAAUAAMAAQAAABQABAAsAAAABgAEAAEAucC6WD//wAA5wLpYP//AAAAAA
BAAYABgAAAAIAAQAAAAAAjgG0ABEAAAAAA8kD8wADAACACwAPABMAFwAbAB8AIwAnACsALwAzADc
AOwBPAGsAACUVizUjFSM1IxUjNSMVIzUjFSM1JRujNSMVIzUjFSM1IxUjNSMVIzUjFSM1IxUjNSM
VIzUjFSM1IxUjNQMVHwYhPwCRITcjDwghNS8HIzUjFSE1IwN2U1NTU1RTU1NTAuxTU1NTVFNTU1M
C7FNTU1NUU1NTU1QCAwUGBgIA0QICACHBQQBAVxsp30ICACHAgUDAQED1AECBAUHBwgIfVP+YFO
zU1NTU1NTU1NTU6dUVFVUFVFRUVRUplNTU1NTU1NTU1P+NgQIBwcGBAMCAQIEBQcHAWgCdPoBAGQ
FAwCHCIF8CQgHBgUEAgFTU1MABAAAAAAD9APeADQAbQCuAQAAAAEfCDc1Lwc1PwIPBy8HHww3HwQ
PATMVPwclLx0jDwMfAgUdAR8GBTUzLxQjDx0BFxUFEDsBPxA1Nyc1LxIPEhUCCg8OGxcVExANCgM
BAQMDCQQDAgECAXESEhMTEhUFRQTFBISEhEHCwYHCAKcKw0NDw8SuQQGAwIBAgRxlgsKCQCGBAM
BAgMDAwUFBQcGBWgICQkKCgsKDAsMDQwNDQ4NDg45BQUADAQEAA/1eAgUGBwkKCwHjeggKDhEUFxs
```

```

1FRMSEA4NCwoJBwcJBjwODg0ODQ0MDQwLDAoLCgoJCQgIBwYHBQUFAwMDAgEBAQECAGYICg0ODxI
SFBUXFwwMDA0MDQwMFxcVFBISDw4MCwgGAgIBAQICAwcJCw0PERITFRUXDAwMDA0NDAwMDAsXFRQ
TEQ8ODQoIBgICAVQEAWgJCgsMCwwbEBAREREZE8VDAwKCgoIBwYFAwIBAQIDBQYHCAoUFQwLCws
LCgoJCACGMwsWFhUVHB3QAQIEBggICgueDg4ODg0NDA0MCwsLCwoKCQgJBwgGBwUFBAQDAwECCw8
NDxETCrIlawSKCAGGBAIB0AoLCwoLCQgNCAkLDA0NDg4ODg4ZFAlBAwMEBAUGBgYIBwkICQoKCws
LDAwMDA0ODQ4ODgH7DQwMDBgWFRQTERAODAoIBgICAQECAGYICgwoEBETFBWGAwMDA0MDQwMCxc
WFRMSEA8NDakHawIBAQEBAQECAwMICwwOEbETFBWfwwMDQAAAAASAN4AAQAAAAAAAAABAAAAQA
AAAAAQASAAEAQAAAAAAAgAHABMAAQAAAAAAAwASABoAAQAAAAABAAASACWAAQAAAAABQALAD4
AAQAAAAABgASAEkAAQAAAAAACgAsAFsAAQAAAAACWASAIcAAwABBakAAAACAjKAawABBakAAQA
kAJsAAwABBakAAgAOAL8AAwABBakAAwAkAM0AAwABBakABAakAPEAAwABBakABQAWARUAawABBak
ABgAkASsAAwABBakACgBYAU8AAwABBakACwAkAacgZWoyLWhlyWYRlcnRlbXBsYXRlUmVndWxhcmV
qMiloZWfKZXJ0ZW1wbGF0ZWVqMiloZWfKZXJ0ZW1wbGF0ZVZlcnNpb24gMS4wZWoyLWhlyWYRlcnR
lbXBsYXRlRm9udCBnZW5lcmF0ZWQgdXNpbmcgU3luY2Zlc2lubiBNZXRYbyBTdHVkaW93d3cuc3l
uY2Zlc2lubi5jb20AIAblAGoAMgAtAGgAZQBhAGQAZQByAHQAZQBtAHAAbABhAHQAZQBSAGUAZWb
lAGwAYQByAGUAagAyAC0AaABlAGEAZABlAHlAdABlAG0AcABsAGEAdABlAGUAagAyAC0AaABlAGE
AZABlAHlAdABlAG0AcABsAGEAdABlAFYAZQByAHMAaQBvAG4AIAAxAC4AMABlAGoAMgAtAGgAZQB
hAGQAZQByAHQAZQBtAHAAbABhAHQAZQBGAG8AbgB0ACAAZwBlAG4AZQByAGEAdABlAGQAIABlAHM
AaQBuAGcAIABTAHkAbgBjAGYAdQBzAGkAbwBuACAATQBlAHQAcgBvACAAUwB0AHUAZABpAG8AdwB
3AHcALgBzAHkAbgBjAGYAdQBzAGkAbwBuAC4AYwBvAG0AAAAAAGAAAAAAAKAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAADAQIBAwEEAAtCVF9DYWxlbmRhcghlbXBsb3llZQAQ)
format('trueType');
        font-weight: normal;
        font-style: normal;
    }
    .e-icon-userlogin:before {
        font-family: 'ej2-headertemplate';
        content: "\e702";
    }
    .e-icon-userlogin {
        margin-right: 5px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <div id="container">
        <div id="Grid"></div>
    </div>
    <script>
        var ele = document.getElementById('container');
        if (ele) {
            ele.style.visibility = "visible";
        }
    </script>
    <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

* The `headerTemplate` property is only applicable to Grid columns that have a header element.

* You can use any HTML or `##Platform_Name##` control in the header template to add additional functionality to the header element.

Stacked header

In Grid, you can group multiple levels of column headers by stacking the Grid columns. This feature allows you to organize the Grid columns in a more structured and understandable way. This can be achieved by setting the [columns->columns](#) property. Within this property, you can define an array of column objects to group together as sub-headers under a main header. You can define the `headerText` property of each sub-header column to set the text for that sub-header.

You can customize the appearance of the stacked header elements by using the `headerTemplate` property. This property accepts an id reference, which allows you to define custom HTML elements or `##Platform_Name##` control to the header element. Here's an example of how to use stacked headers with a custom `headerTemplate` in Syncfusion Grid.

INDEX.JS

```
var grid = new ej.grids.Grid({
  dataSource: data,
  allowPaging: true,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', width: 120, textAlign:
    'Center', headerTemplate: '#orderID' },
    {
      headerText: 'Order Details', headerTemplate: '#orderDate',
      columns: [
        { field: 'OrderDate', headerText: 'Order Date', format:
        'yMd', width: 130, textAlign: 'Right', minWidth: 10, },
        { field: 'Freight', headerText: 'Freight($)', width: 120,
        format: 'C1', textAlign: 'Right', minWidth: 10, },
      ]
    },
    {
      headerText: 'Ship Details',
      headerTemplate: '<div> <span>Ship Details</span><span><i
class="fa fa-truck"></i></span></div>',
      columns: [
        { field: 'ShippedDate', headerText: 'Shipped Date', format:
        'yMd', textAlign: 'Right', width: 150, minWidth: 10, },
        { field: 'ShipCountry', headerText: 'Ship Country', width:
        150, minWidth: 10, },
      ]
    }
  ]
});
grid.appendTo('#Grid');
var dropdownData = ['Order Details', 'Order Information'];
var dropdownList = new ej.dropdowns.DropDownList({
  value: 'Order Details',
  popupHeight: '200px',
  dataSource: dropdownData,
});
dropdownList.appendTo('#orderdate');
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
```

```

<title>EJ2 Grid</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-
awesome.min.css">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
<div id="container">
<div id="Grid"></div>
</div>
<script>
var ele = document.getElementById('container');
if (ele) {
ele.style.visibility = "visible";
}
</script>
<script id="orderID" let-data>
<a href="#">OrderID</a>
</script>
<div id="orderDate" let-data>
<div id="orderdate"></div>
</div>
<script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Align the text of header text

You can horizontally align the text in column headers of the Grid control using the [headerTextAlign](#) property. By default, the text is aligned to the left, but you can change the alignment by setting the value of the `headerTextAlign` property to one of the following options:

- **Left:** Aligns the text to the left (default).
- **Center:** Aligns the text to the center.
- **Right:** Aligns the text to the right.
- **Justify:** Header text is justified.

Here is an example of using the `headerTextAlign` property to align the text of a Grid column header:

INDEX.JS

```
var grid = new ej.grids.Grid({
  dataSource: data,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', width: 120 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 140 },
    { field: 'Freight', headerText: 'Freight', format: 'C', width: 120 },
    { field: 'OrderDate', headerText: 'Order Date', format: 'yMd', width:
140 },
  ],
  height: 315,
});
grid.appendTo('#Grid');
var alignmentData = [
  { text: 'Left', value: 'Left' },
  { text: 'Right', value: 'Right' },
  { text: 'Center', value: 'Center' },
  { text: 'Justify', value: 'Justify' }
]
var dropdownList = new ej.dropdowns.DropDownList({
  value: 'Left',
  popupHeight: '240px',
  width: 100,
  dataSource: alignmentData,
  change: changeAlignment,
});
dropdownList.appendTo('#dropdown');
function changeAlignment(args) {
  grid.columns.forEach((column) => {
    column.headerTextAlign = args.value
  })
  grid.refreshHeader();
}
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div style="padding-bottom: 10px">
    <br>
    <label 30px 17px 0 0>Align the text of header text :</label>
    <input type="text" tabindex="1" id="dropdown" />
  </div>
  <div id="Grid">
  </div>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

* The `headerTextAlign` property only changes the alignment of the text in the column header, and not the content of the column. If you want to align both the column header and content, you can use the [textAlign](#) property.

* You can also use the `headerTextAlign` property with the stacked header feature in Syncfusion Grid. The property will align the header text in the sub-headers as well.

Autowrap the header text

The autowrap allows the cell content of the grid to wrap to the next line when it exceeds the boundary of the specified cell width. The cell content wrapping works based on the position of white space between words. To support the Autowrap functionality in Syncfusion Grid, you should set the

appropriate [width](#) for the columns. The column width defines the maximum width of a column and helps to wrap the content automatically.

To enable autowrap, set the `allowTextWrap` property to **true**. You can also configure the auto wrap mode by setting the `textWrapSettings.wrapMode` property.

Grid provides the below three options for configuring:

- **Both:** This is the default value for `wrapMode`. With this option, both the grid header text and content is wrapped.
- **Header:** With this option, only the grid header text is wrapped.
- **Content:** With this option, only the grid content is wrapped.

* If a column width is not specified, then the Autowrap of columns will be adjusted with respect to the grid's width.

* If a column's header text contains no white space, the text may not be wrapped.

* If the content of a cell contains HTML tags, the Autowrap functionality may not work as expected. In such cases, you can use the [headerTemplate](#) and [template](#) properties of the column to customize the appearance of the header and cell content.

In the example below, the `textWrapSettings.wrapMode` property is set to **Header** only the grid header text is wrap to the next line.

INDEX.JS

```
var grid = new ej.grids.Grid({
  dataSource: data,
  allowPaging: true,
  allowTextWrap: true,
  textWrapSettings: { wrapMode: 'Header' },
  height: 400,
  columns: [
    { field: 'Inventor', headerText: 'Inventor Name', width: 180, textAlign: 'Right' },
    { field: 'NumberofPatentFamilies', headerText: 'Number of Patent Families', width: 180, textAlign: "Right" },
    { field: 'Country', headerText: 'Country', width: 140, textAlign: 'Right' },
    { field: 'Active', headerText: 'Active', width: 120 },
    { field: 'Mainfieldsofinvention', headerText: 'Main Feilds Of Invention', width: 200 },
  ],
});
grid.appendTo('#Grid');
var dropdownData = [
  { text: 'Header', value: 'Header' },
  { text: 'Both', value: 'Both' },
]
var dropdown = new ej.dropdowns.DropDownList({
  dataSource: dropdownData,
  popupHeight: '240px',
  width: '100px',
  value: grid.textWrapSettings.wrapMode,
  change: valueChange,
```



```
});
dropdown.appendTo('#dropdownlist');
function valueChange(args) {
    grid.textWrapSettings.wrapMode = args.value;
}
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <div style="padding-bottom: 10px">
        <label padding: 30px 17px 0 0>Autowrap for header column :</label>
        <input type="text" tabindex="1" id="dropdownlist" />
    </div>
    <div id="container">
        <div id="Grid"></div>
    </div>
    <script>
        var ele = document.getElementById('container');
        if (ele) {
            ele.style.visibility = "visible";
        }
    </script>
```

```

</script>
<script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Change the height of header

Changing the height of the header can be useful in cases where the default height is not sufficient to display the header content cell. For example, if you have a header with a lot of text or if you want to add an image to the header, you may need to increase the height of the header to accommodate the content. This can be easily achieved by changing the height of the header using CSS or by dynamically adjusting the height using a methods.

Using CSS

You can use CSS to override the default height of the **.e-grid .e-headercell** class to change the height of the header. Here is an example code snippet:

```

`css
.e-grid .e-headercell {
height: 130px;
}
`

```

Using methods

To change the height of the header dynamically, you can use the [getHeaderContent](#) method to get the header content element of the Syncfusion Grid. Then, you can use the **querySelectorAll** method to get all the header cell elements with the class **e-headercell**. Finally, you can loop through each header cell element and set its style property to adjust the height.

INDEX.JS

```

var grid = new ej.grids.Grid({
  dataSource: data,
  columns: [
    {field: 'OrderID', headerText: 'Order ID', width: 120},
    {field: 'CustomerID', headerText: 'Customer ID', width: 140},
    {field: 'Freight', headerText: 'Freight', format: 'C', width: 120},
    {field: 'OrderDate', headerText: 'Order Date', format: 'yMd', width: 140},
  ],
});
grid.appendTo('#Grid');
var button = new ej.buttons.Button({
  content: 'CHANGE HEIGHT TO 20PX',
});
button.appendTo('#small');
var button1 = new ej.buttons.Button({
  content: 'DEFAULT HEIGHT TO 42PX',
});
button1.appendTo('#medium');
var button2 = new ej.buttons.Button({
  content: 'CHANGE HEIGHT TO 60PX',
});

```

```

button2.appendTo('#big');
document.getElementById('changeHeight').onclick = function (event) {
    var heightMap = { small: '20px', medium: '42px', big: '60px' };
    var headerCells = (grid).getHeaderContent().querySelectorAll('.e-
headercell');
    headerCells.forEach((headerCell) => {
        headerCell.style.height = (heightMap)[
            event.target.id
        ];
    });
}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <div id="changeHeight">
        <button ej-button id="small" cssClass="e-small">
            Change height to 20px
        </button>
        <button ej-button id="medium" cssClass="e-small">
            Default height to 40px
    </div>

```

```

    </button>
    <button ejs-button id="big" cssClass="e-small">
      Change height to 60px
    </button>
  </div>
  <br>
  <div id="container">
    <div id="Grid"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

* You can also use the [getHeaderTable](#) method to get the table element of the header, and then adjust the height.

* You cannot change the height of row below the default height of 42px using the **e-columnheader** class.

Change the header text dynamically

The Syncfusion Grid controls provides a way to modify the header text of a corresponding column in real-time based on events or other events. This feature can be useful in various scenarios, such as displaying a custom header text for a specific column or updating the header text dynamically based on input. By allowing for dynamic changes to the header text, the Grid provides a more flexible and customizable experience.

Using event

To modify the header text of a corresponding column dynamically, you can use the [headerCellInfo](#) event provided by the Syncfusion Grid. This event is triggered for each header cell element rendered in the Grid.

When the [headerCellInfo](#) event is triggered, it provides a **HeaderCellInfoEventArgs** object as a parameter. This object contains the following properties:

- **cell**: Defines the header cell that is being modified.
- **node**: Defines the DOM element of the header cell that is being modified.

You can use these properties to access and modify the header text of the corresponding column. Once the header text is modified, you can refresh the Grid to reflect the changes by calling the [refreshHeader](#) method of the Grid.

Using method

The Grid control provides several methods that allow you to change the column header text dynamically. Here are some of the methods you can use:

1. [getColumnByField](#): This method takes a field name as a parameter and returns the entire column object that corresponds to that field name, including properties such as headerText, width, and alignment. You can use this method to modify any aspect of the column. 2. [getColumnHeaderByField](#): Retrieves the header element of a column based on its field name. You can modify the **textContent** property of the header element to change the header text. This method does not return a reference to the column object itself, only to the header element. 3. [getColumnIndexByField](#): Retrieves the index of a column based on its field name. You can then use the [getColumnByIndex](#) method to retrieve the column object and modify its **headerText** property to change the header text. 4. [getColumnByUid](#): Retrieves the column object based on its unique identifier (UID). You can modify the **headerText** property of the column object to change the header text. 5. [getColumnHeaderByIndex](#): Retrieves the header element of a column based on its zero-based index. You can modify the **textContent** property of the header element to change the header text. This method does not return a reference to the column object itself, only to the header element. 6. [getColumnIndexByUid](#): Retrieves the index of a column based on its unique identifier (UID). You can then use the [getColumnByIndex](#) method to retrieve the column object and modify its **headerText** property to change the header text. 7. [getColumnHeaderByUid](#): Retrieves the header element of a column based on its unique identifier (UID). You can modify the **textContent** property of the header element to change the header text. This method does not return a reference to the column object itself, only to the header element. If you only have an **template** for the column header, and the column itself is not defined with a **field**, then you can use the [getColumnHeaderByUid](#) method to get a reference to the header element and modify its text content to change the header text.

* When you change the header text dynamically, make sure to **refresh** the Grid to reflect the changes by calling the [refreshHeader](#) method.

* The UID is automatically generated by the Grid control and may change whenever the grid is refreshed or updated.

Here is an example of how to change the header text of a column using the [getColumnByField](#) method:

INDEX.JS

```
var grid = new ej.grids.Grid({
  dataSource: data,
  allowPaging: true,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', width: 120 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 140 },
    { field: 'Freight', headerText: 'Freight', format: 'C', width: 120 },
    { field: 'OrderDate', headerText: 'Order Date', format: 'yMd', width:
140 },
  ],
});
grid.appendTo('#Grid');
var columns =
[
  { text: 'OrderID', value: 'OrderID' },
  { text: 'CustomerID', value: 'CustomerID' },
  { text: 'Freight', value: 'Freight' },
  { text: 'OrderDate', value: 'OrderDate' },
]
var field = { text: 'text', value: 'value' };
var dropdown = new ej.dropdowns.DropDownList({
```

```

    dataSource: columns,
    fields: field,
    value: 'OrderID',
    popupHeight: '240px',
    width: 220,
  });
  dropdown.appendTo('#dropdownlist');
  var textbox = new ej.inputs.TextBox({
    placeholder: 'Enter new header text',
    width: 220,
  });
  textbox.appendTo('#textboxvalue');
  var button = new ej.buttons.Button({
    content: 'Change',
  });
  button.appendTo('#buttons');
  document.getElementById('buttons').onclick = function () {
    if (textbox.value.trim() !== '') {
      var column = grid.getColumnByField(dropdown.value);
      column.headerText = textbox.value;
      grid.refreshHeader();
    }
  };
};

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
<div id="container">
<div style="padding-bottom: 10px">
<label padding: 30px 17px 0 0>Select a column Name : </label>
<input type="text" tabindex="2" id="dropdownlist" />
</div>
<div style="padding-bottom: 10px">
<label padding: 30px 17px 0 0>Enter New Header Text : </label>
<input type="text" tabindex="2" id="textboxvalue" />
</div>
<div style="padding-bottom: 10px">
<label padding: 30px 17px 0 0>Click the change button : </label>
<button ej-button id="buttons" cssClass="e-small"></button>
</div>
<div id="Grid"></div>
</div>
<script>
var ele = document.getElementById('container');
if (ele) {
ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Changing header text using headerValueAccessor property

The [headerValueAccessor](#) property in Syncfusion Grid allows you to customize the text of a column header cell, which can be useful in scenarios where you want to change the text to display it in a different language, format or add additional information to the header. This property is triggered every time the header cell is rendered.

To enable the `headerValueAccessor` property, you need to set the `headerValueAccessor` property of the corresponding column. This property accepts a callback function that takes two arguments:

- **field:** Represents the current field of the column.
- **column:** Represents the current column object.

* The `headerValueAccessor` property should only be used to change the text of the header and not to perform any DOM-oriented operations such as adding or manipulating DOM elements in the header. In such cases, you should use the [headerCellInfo](#) event instead.

* The `headerValueAccessor` property is triggered every time the header cell is rendered or refreshed.

* The callback function defined for the `headerValueAccessor` property should return a string that represents the new text of the column header.

* If you only need to refresh the column header, you can dynamically change the header content using the [refreshHeader](#) method.

* You can use this property for individual columns or for all columns by adding it to the grid's properties.

Here's an example of how to use the `headerValueAccessor` property to change the header text of a column:

INDEX.JS

```
var grid = new ej.grids.Grid({
  dataSource: data,
  allowPaging: true,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', headerValueAccessor:
headerValueAccessor, width: 120 },
    { field: 'CustomerID', headerText: 'Customer ID', headerValueAccessor:
headerValueAccessor, width: 140 },
    { field: 'Freight', headerText: 'Freight', headerValueAccessor:
headerValueAccessor, format: 'C', width: 120 },
    { field: 'OrderDate', headerText: 'Order Date', headerValueAccessor:
headerValueAccessor, format: 'yMd', width: 140 },
  ],
});
grid.appendTo('#Grid');
var columns = [
  { text: 'OrderID', value: 'OrderID' },
  { text: 'CustomerID', value: 'CustomerID' },
  { text: 'Freight', value: 'Freight' },
  { text: 'OrderDate', value: 'OrderDate' },
];
var field = { text: 'text', value: 'value' };
var dropdown = new ej.dropdowns.DropDownList({
  dataSource: columns,
  fields: field,
  value: 'OrderID',
  popupHeight: '240px',
  width: 220,
});
dropdown.appendTo('#dropdownlist');
var textbox = new ej.inputs.TextBox({
  placeholder: 'Enter new header text',
  width: 220,
});
textbox.appendTo('#textboxvalue');
var button = new ej.buttons.Button({
  content: 'Change',
});
button.appendTo('#buttons');
function headerValueAccessor(field, columns) {
  if (textbox && textbox.value && textbox.value.trim() !== '' &&
columns.field === dropdown.value) {
    columns.headerText = textbox.value;
  }
}
document.getElementById('buttons').onclick = function () {
  grid.refreshHeader();
}
```



```
};
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container">
    <div style="padding-bottom: 10px">
      <label padding: 30px 17px 0 0>Select a column Name : </label>
      <input type="text" tabindex="2" id="dropdownlist" />
    </div>
    <div style="padding-bottom: 10px">
      <label padding: 30px 17px 0 0>Enter New Header Text : </label>
      <input type="text" tabindex="2" id="textboxvalue" />
    </div>
    <div style="padding-bottom: 10px">
      <label padding: 30px 17px 0 0>Click the change button : </label>
      <button ej-s-button id="buttons" cssClass="e-small"></button>
    </div>
    <div id="Grid"></div>
  </div>
<script>
```

```

var ele = document.getElementById('container');
if (ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Changing the header text of all columns

If you want to change the header text of all columns in the grid, you can loop through the Columns collection of the grid and set the `headerText` property for each column. Here is an example:

INDEX.JS

```

var grid = new ej.grids.Grid({
    dataSource: data,
    columns: [
        { field: 'OrderID', headerText: 'OrderID', textAlign: 'Right', width: 90 },
        { field: 'CustomerID', headerText: 'CustomerID', width: 120 },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right', format: 'C2', width: 90 },
        { field: 'ShipCity', headerText: 'Ship City', format: 'yMd', width: 120 }
    ],
    height: 280,
});
grid.appendTo('#Grid');
var headerTextMap =
{
    'OrderID': 'Order ID',
    'CustomerID': 'Customer ID',
    'Freight': 'Freight Charge',
    'ShipCity': 'Ship To City'
}
var button = new ej.buttons.Button({
    content: 'Change Header Text',
    cssClass: "e-success",
});
button.appendTo('#buttons');
document.getElementById('buttons').onclick = function () {
    grid.columns.forEach((column) => {
        column.headerText = headerTextMap[column.field];
    });
    grid.refreshHeader();
};

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">

```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container">
    <button ej-button id="buttons" cssClass="e-success"></button>
    <br><br>
    <div id="Grid"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Change the orientation of header text

By default, the text in the column headers of the Syncfusion Grid controls is oriented horizontally. However, in some cases, you may want to change the orientation of the header text to vertical, diagonal, or at a custom angle. This can be achieved by adding a custom CSS class to the column header cell using the [customAttributes](#) property of the Grid columns.

Follow the below steps to change the orientation of the header text in Grid:

Step 1: Create a CSS class with orientation style for grid header cell

To **rotate** the header text, you can create a CSS class with the **transform** property that rotates the header text 90 degrees. This class will be added to the header cell using the **customAttributes** property.

```
`css
.orientationcss .e-headercelldiv {
transform: rotate(90deg);
}
`
```

Step 2: Add the custom CSS class to the grid column

Once you have created the CSS class, you can add it to the particular column by using the **customAttributes** property. This property allows you to add any custom attribute to the grid column.

For example, to add the orientationcss class to the Freight column, you can use the following code:

```
{ field: 'Freight', headerText: 'Freight', textAlign: 'Center', format: 'C2', customAttributes: {class:
'orientationcss'}, width: 80}
`
```

Step 3: Resize the header cell height

After adding the custom CSS class to a column, you need to resize the header cell height so that the rotated header text is fully visible. You can do this by using the following code:

```
setHeaderHeight(args) {
let textWidth: number = document.querySelector('.orientationcss > div').scrollWidth;//Obtain the width
of the headerText content.
let headerCell: NodeList = document.querySelectorAll('.e-headercell');
for(let i: number = 0; i < headerCell.length; i++) {
(<HTMLElement>headerCell.item(i)).style.height = textWidth + 'px'; //Assign the obtained textWidth as
the height of the headerCell.
}
}
`
```

INDEX.JS

```
var customAttributes = { class: 'orientationcss' };
var grid = new ej.grids.Grid({
  dataSource: data,
  created: setHeaderHeight,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right', width:
100 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
    { field: 'Freight', textAlign: 'center', format: 'C2', customAttributes:
customAttributes, width: 80 },
```

```

    { field: 'ShipCity', headerText: 'Ship City', width: 100, format: 'yMd'
  },
  ],
  height: 240,
});
grid.appendTo('#Grid');
function setHeaderHeight() {
  var textWidth = document.querySelector('.orientationcss >
div').scrollWidth;
  var headerCell = document.querySelectorAll('.e-headercell');
  for (var i = 0; i < headerCell.length; i++) {
    // Assign the obtained textWidth as the height of the headerCell.
    headerCell.item(i).style.height = textWidth + 'px';
  }
}
}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <style>
    body {
      touch-action: none;
    }
    .orientationcss .e-headercelldiv {
      transform: rotate(90deg);
    }
  </style>

```

```

    }
  </style>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container">
    <div id="Grid"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Custom tooltip for header

Custom tooltips for headers provide additional information when hovering over a column header in the Syncfusion Grid. This can be useful in situations where there is not enough space to display all of the information related to a column, or when there is additional context that may be helpful.

To enable custom tooltips for headers, you can use the [beforeRender](#) event of the Grid control. This event is triggered for each header cell before it is rendered, allowing you to add a custom tooltip to the header cell using [tooltip](#) control.

Here's an example of how to use the `beforeRender` event to add a custom tooltip to a header cell:

INDEX.JS

```

var grid = new ej.grids.Grid({
  dataSource: data,
  allowPaging: true,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', width: 120 },
    { field: 'Freight', headerText: 'Freight', width: 120, format: 'C2' },
    { field: 'ShipName', headerText: 'Ship Name', width: 150 },
    { field: 'ShipCountry', headerText: 'Ship Country', width: 120 },
    { field: 'OrderDate', headerText: 'Order Date', type: 'date', format:
'yMd', width: 120 },
  ],
});
grid.appendTo('#Grid');
var columnDescriptions = {
  'Order ID': 'A unique number assigned to each order.',
  'Freight': 'The cost of shipping the order.',
  'Ship Name':
    'The name of the person or company who will receive the shipment.',
  'Ship Country': 'The country to which the shipment will be sent.',
  'Order Date': 'The date when the order was placed.',
};
var tooltip = new ej.popups.Tooltip({
  beforeRender: beforeRender,
  target: '.e-headertext',

```

```
});
tooltip.appendTo('#tooltip');
function beforeRender(args) {
    var description = columnDescriptions[args.target.innerText];
    if (description) {
        tooltip.content = args.target.innerText + ': ' + description;
    }
}
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <div id="container">
        <div id="tooltip">
            <div id="Grid"></div>
        </div>
    </div>
    <script>
        var ele = document.getElementById('container');
        if (ele) {
            ele.style.visibility = "visible";
```

```

    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

* The [headerCellInfo](#) event can also be used to customize the header tooltip. This event is triggered for each header cell after it is rendered.

Customize header text styles

Customizing the grid header styles allows you to modify the appearance of the column header in the Grid control to meet your design requirements. You can customize the font, background color, and other styles of the header cells. To customize the header styles in the grid, you can use CSS, properties, methods, or event support provided by the Syncfusion `##Platform_Name##` Grid control.

Using CSS

You can apply styles to the header cells using CSS selectors. The Grid provides a class name for each header cell element, which you can use to apply styles to that specific header cell. The `.e-headercell` class can be used to change the background color and text color of the column header.

`CSS

```

.e-grid .e-headercell {
background-color: #a2d6f4;
color:rgb(3, 2, 2);
}
`

```

Here's an example that demonstrates how to customize the appearance of a specific column header in the Grid using `className`.

INDEX.JS

```

var grid = new ej.grids.Grid({
  dataSource: data,
  allowPaging: true,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', width: 120, textAlign: 'Right' },
    { field: 'CustomerID', headerText: 'Customer ID', width: 150, textAlign: 'Right' },
    { field: 'OrderDate', headerText: 'Order Date', width: 130, format: 'yMd', textAlign: 'Right' },
    { field: 'Freight', headerText: 'Freight', width: 140, format: 'C2', textAlign: 'Right' },
    { field: 'ShipCountry', headerText: 'Ship Country', width: 130, format: 'yMd', textAlign: 'Right' },
  ]
});
grid.appendTo('#Grid');

```

INDEX.HTML


```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <style>
    .e-grid .e-headercell {
      background-color: #a2d6f4;
      color: rgb(3, 2, 2);
    }
  </style>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container">
    <div id="Grid"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Using property

You can customize the appearance of the column headers in Grid using the [customAttributes](#) property. The `customAttributes` property takes an object with the name-value pair to customize the CSS properties for grid header cells. You can also set multiple CSS properties to the custom class using the `customAttributes` property.

To customize the header of a column, you can follow the steps below:

Step 1: Define a CSS class that specifies the styles you want to apply to the header cell of the column. For example, to change the background color and text color of the header cell, define a CSS class like this:

```
`CSS
.e-grid .e-headercell.customcss {
background-color: rgb(43, 205, 226);
color: black;
}
```

Step 2: Set the `customAttributes` property of the desired column to an object that contains the CSS class `customcss`. This CSS class will be applied to the header cell of the specified column in the Grid.

```
`ts
{field: "Freight" headerText: "Freight" customAttributes: {class: '.customcss'}}
`
```

The following example demonstrates how to customize the appearance of the **OrderID** and **Freight** columns using custom attributes.

INDEX.JS

```
var grid = new ej.grids.Grid({
  dataSource: data,
  allowPaging: true,
  columns: [
    {field: 'OrderID',headerText: 'Order ID',customAttributes: { class:
'customcss' },textAlign: 'Center'},
    { field: 'CustomerName', headerText: 'Customer Name', textAlign:
'Center'},
    {field: 'Freight',headerText: 'Freight',customAttributes: { class:
'customcss' },textAlign: 'Center'},
    {field: 'OrderDate',headerText: 'Order Date',format: 'yMd',textAlign:
'Center'}
  ],
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
```

```

<title>EJ2 Grid</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<style>
.e-grid .e-headercell.customcss {
    background-color: rgb(43, 205, 226);
    color: black;
}
</style>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
<div id="container">
<div id="Grid"></div>
</div>
<script>
var ele = document.getElementById('container');
if (ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Using method

The Syncfusion Grid provides methods to customize the appearance of the grid columns header.

1. [getColumnHeaderByIndex](#): The method is used to customize the appearance of a specific column header in the grid by specifying the index of the column for which you want to customize the header.
2. [getColumnHeaderByField](#): This method is used to retrieve the header element of a specific column by its field name. You can use the retrieved element to customize the appearance of the header element.
3. [getColumnHeaderByUid](#): This method is used to retrieve the header element of a specific column by its unique ID. You can use the retrieved element to customize the appearance of the header element.
4. [getColumnIndexByField](#): This method is used to retrieve the index of a specific column by its field name. You can use the retrieved index to access the header element and customize its appearance.
5. [getColumnIndexByUid](#): This method is used to retrieve the index of a specific column by its unique ID. You can use the retrieved index to access the header element and customize its appearance.

Here's an example of how to use these methods to change the style of a specific column header:

INDEX.JS

```
var grid = new ej.grids.Grid({
  dataSource: data,
  allowPaging: true,
  dataBound: dataBound,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', width: 120, textAlign: 'Right' },
    { field: 'CustomerName', headerText: 'Customer Name', width: 150 },
    { field: 'Freight', headerText: 'Freight', width: 120, format: 'C2',
      textAlign: 'Right' },
    { field: 'ShipCountry', headerText: 'Ship Country', width: 150 },
  ],
  height: 315,
});
grid.appendTo('#Grid');
function dataBound() {
  grid.getColumnHeaderByIndex(0).style.color = '#0d0b0b';
  grid.getColumnHeaderByField('CustomerName').style.background = '#f45ddeab';
  grid.getColumnHeaderByField('CustomerName').style.color = '#0d0b0b';
  grid.getColumnHeaderByUid('grid-column2').style.background = '#f45ddeab';
  var columnIndex = grid.getColumnIndexByField('ShipCountry');
  grid.getColumnHeaderByIndex(columnIndex).style.color = '#0d0b0b';
  var index = grid.getColumnIndexByUid('grid-column2');
  grid.getColumnHeaderByIndex(index).style.color = '#0d0b0b';
}
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container">
    <div id="Grid"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

* The UID is automatically generated by the Grid control and may change whenever the grid is refreshed or updated.

Using event

To customize the appearance of the grid header, you can handle the [headerCellInfo](#) event of the grid. This event is triggered when each header cell is rendered in the grid, and provides an object that contains information about the header cell. You can use this object to modify the styles of the header column.

The following example demonstrates how to add a `headerCellInfo` event handler to the grid. In the event handler, checked whether the current header column is **Order Date** field and then applied the appropriate CSS class to the cell based on its value.

INDEX.JS

```
var grid = new ej.grids.Grid({
  dataSource: data,
  allowPaging: true,
  headerCellInfo: onHeaderCellInfo,
  columns: [
    {field: 'OrderID', headerText: 'Order ID', width: 120, textAlign:
'Center'},
    {field: 'CustomerID', headerText: 'CustomerID', width: 150 },
    {field: 'OrderDate', headerText: 'Order Date', width: 130, format:
'yMd', textAlign: 'Center'},
    { field: 'Freight', headerText: 'Freight', width: 120, format:
'C2', textAlign: 'Center' },
    {field: 'ShippedDate', headerText: 'Shipped Date', width: '130', format:
'yMd', textAlign: 'Center'}
  ],
});
grid.appendTo('#Grid');
function onHeaderCellInfo(args) {
  if (args.cell.column.field == 'OrderDate') {
    args.node.classList.add('customcss');
  }
}
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<style>
.e-grid .e-headercell.customcss {
  background-color: rgb(43, 205, 226);
  color: black;
}
</style>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
<div id="container">
<div id="Grid"></div>
</div>
<script>
var ele = document.getElementById('container');
if (ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body>
</html>

```

How to refresh header

The refresh header feature in the Syncfusion `##Platform_Name##` Grid allows you to update the header section of the grid whenever changes are made to the grid's columns. This feature is useful when you want to reflect changes in the header immediately, such as modifying the column header text, width, or alignment.

To use the refresh header feature, you can call the [refreshHeader](#) method of the Grid control. This method updates the grid header with the latest changes made to the columns.

The following example demonstrates how to use the `refreshHeader` method to update the grid header:

INDEX.JS

```

var grid = new ej.grids.Grid({
  dataSource: data,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', width: 100, },
    { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
    { field: 'OrderDate', headerText: 'Order Date', format: 'yMd' },
    { field: 'Freight', headerText: 'Freight', width: 120 }
  ],
  height: 280,
});
grid.appendTo('#Grid');
var button = new ej.buttons.Button({
  content: 'Refresh Header',

```

```
});
button.appendTo('#buttons');
document.getElementById('buttons').onclick = function () {
    var column = grid.getColumnByIndex(1);
    column.headerText = 'New Header Text'; // update the header text of the
    column object
    grid.refreshHeader();
};
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <div id="container">
        <div style="padding-bottom: 10px">
            <button ej2-button id="buttons" cssClass="e-small"></button>
            <br><br>
            <div id="Grid"></div>
        </div>
        <script>
            var ele = document.getElementById('container');
            if (ele) {
```



```

        ele.style.visibility = "visible";
    }
</script>
<script src="index.js" type="text/javascript"></script>
</body>
</html>

```

* The `refreshHeader` method updates only the grid header and not the entire grid.

* If you want to refresh the entire grid, you can use the `refresh` method instead.

How to get header element

To get the header element in a Syncfusion Grid, you can use one of the following methods:

1. [getHeaderContent](#): This method returns the header div element of the Grid. You can use this method to access the entire header content of the Grid.

```

`ts
const headerElement = grid.getHeaderContent();
`

```

2. [getHeaderTable](#): This method returns the header table element of the Grid. You can use this method to access only the header table of the Grid.

```

`ts
const headerTableElement = grid.getHeaderTable();
`

```

3. [getColumnHeaderByUid](#): This method returns the column header element by its unique identifier.

```

`ts
const columnHeaderElement = grid.getColumnHeaderByUid("e-grid2");
`

```

4. [getColumnHeaderByIndex](#): This method returns the column header element by its index.

```

`ts
const columnHeaderElement = grid.getColumnHeaderByIndex(0);
`

```

5. [getColumnHeaderByField](#): This method returns the column header element by its field name.

```

`ts
const columnHeaderElement = grid.getColumnHeaderByField("OrderID");
`

```

* The UID is automatically generated by the Grid control and may change whenever the grid is refreshed or updated.

Column template in ##Platform_Name## Grid control

Render image in a column

The column [template](#) has options to display custom element instead of a field value in the column.

INDEX.JS

```
var grid = new ej.grids.Grid({
  dataSource: employeeData,
  columns: [
    {
      headerText: 'Employee Image', textAlign: 'Center',
      template: '#template', width: 150
    },
    { field: 'EmployeeID', headerText: 'Employee ID', textAlign:
'Right', width: 125 },
    { field: 'FirstName', headerText: 'Name', width: 120 },
    { field: 'Title', headerText: 'Title', width: 170 }
  ],
  height: 315
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <script id="template" type="text/x-template">
            <div class="image">
                
            </div>
        </script>
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Grid actions such as editing, grouping, filtering and sorting etc. will depend upon the column [field](#). If the [field](#) is not specified in the template column, the grid actions cannot be performed.

Render other components in a column

You can render any component in a grid column using the [template](#) property.

To render other components in the grid, ensure the following steps:

Step 1:

Initialize the column template for your custom component.

```

`ts
template: `<div>
<select class="e-control e-dropdownlist">
<option value="1" selected="selected">Order Placed</option>
<option value="2">Processing</option>
<option value="3">Delivered</option>
</select>
</div>`
`

```

Step 2:

Using the [queryCellInfo](#) event, you can render the DropDown component with the following code.

```
`ts
function dropdown(args: QueryCellInfoEventArgs) {
let ele=args.cell.querySelector('select');
let drop = new DropDownList({popupHeight: 150, popupWidth: 150});
drop.appendTo(ele);
}
`
```

INDEX.JS

```
var grid = new ej.grids.Grid({
  dataSource: data,
  columns: [
    {
      headerText: 'Order Status',
      template:
        `<div>
          <select class="e-control e-dropdownlist">
            <option value="1" selected="selected">Order
Placed</option>
            <option value="2">Processing</option>
            <option value="3">Delivered</option>
          </select>
        </div>`, width: 140
    },
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
    { field: 'OrderDate', headerText: 'Order Date', width: 100, format:
'yMd' },
    { field: 'ShipCity', headerText: 'Ship City', width: 100 },
  ],
  height: 315,
  queryCellInfo: dropdown
});
grid.appendTo('#Grid');
function dropdown(args) {
  var ele = args.cell.querySelector('select');
  var drop = new ej.dropdowns.DropDownList({ popupHeight: 150, popupWidth:
150 });
  drop.appendTo(ele);
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Using condition template

You can render the template elements based on condition.

In the following code, checkbox is rendered based on **Discontinued** field value.

```

<script id="template" type="text/x-template">
<div class="template_checkbox">

```

```

${if(Discontinued)}
<input type="checkbox" checked> ${else}
<input type="checkbox"> ${/if}
</div>
</script>
`

```

INDEX.JS

```

var grid = new ej.grids.Grid({
  dataSource: productData,
  columns: [
    {
      headerText: 'Discontinued', textAlign: 'Center',
      template: '#template', width: 120
    },
    { field: 'ProductID', headerText: 'Product ID', textAlign:
'Right', width: 100 },
    { field: 'ProductName', headerText: 'Name', width: 120 },
    { field: 'SupplierID', headerText: 'SupplierID', width: 100 },
    { field: 'UnitsInStock', headerText: 'Stock', width: 100,
  textAlign: 'Right' }
  ],
  height: 315
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-grids/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <script id="template" type="text/x-template">
            <div class="template_checkbox">
                ${if(Discontinued)}
                <input type="checkbox" checked> ${else}
                <input type="checkbox"> ${/if}
            </div>
        </script>
        <div id="Grid"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

How to get the row object by clicking on the template element

You can get the row object without selecting the row and achieve it using the column template feature and the `getRowObjectFromUID` method of the Grid.

In the following sample, the button element is rendered in the Employee Data column. By clicking the button, you can get the row object using the `getRowObjectFromUID` method of the Grid and display it in the console.

INDEX.JS

```

var grid = new ej.grids.Grid({
    dataSource: employeeData,
    columns: [
        {
            headerText: 'Employee Data', textAlign: 'Right',
            template: '#template', width: 150, isPrimaryKey: true
        },
        { field: 'EmployeeID', headerText: 'Employee ID', textAlign:
'Right', width: 130 },
        { field: 'FirstName', headerText: 'Name', width: 120 },
        { field: 'Title', headerText: 'Title', width: 170 }
    ],
    height: 315,

```

```

        recordClick: (args) => {
            if (args.target.classList.contains('empData')) {
                var rowObj =
grid.getRowObjectFromUID(ej.base.closest(args.target, '.e-
row').getAttribute('data-uid')
            );
            console.log(rowObj);
        }
    }
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```



```

<div id="container">
  <script id="template" type="text/x-template">
    <button class="empData">Employee Data</button>
  </script>
  <div id="Grid"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Complex data binding in ##Platform_Name## Grid control

You can achieve complex data binding in the grid by using the dot(.) operator in the [column.field](#).

INDEX.JS

```

var grid = new ej.grids.Grid({
  dataSource: complexData,
  columns: [
    { field: 'EmployeeID', headerText: 'Employee ID', textAlign:
'Right', width: 100 },
    { field: 'Name.FirstName', headerText: 'First Name', width: 120 },
    { field: 'Name.LastName', headerText: 'Last Name', width: 100 },
    { field: 'Title', headerText: 'Title', width: 150 }
  ],
  height: 315
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

For OData and ODataV4 adaptors, you need to add [expand](#) query to the [query](#) property (of Grid), to eager load the complex data.

```
`ts
```

```

import { DataManager, ODataAdaptor, Query } from '@syncfusion/ej2-data';

let data: DataManager = new DataManager({
    adaptor: new ODataAdaptor(),
    crossDomain: true,
    url: 'https://js.syncfusion.com/demos/ejServices/Wcf/Northwind.svc/Orders'
});

let query: Query = new Query().expand('Employee');

let grid: Grid = new Grid({
    dataSource: data,
    query: query,
    columns: [

```

```
{ field: 'OrderID', headerText: 'Order ID', textAlign: 'Right', width: 100 },
{ field: 'CustomerID', headerText: 'Customer ID', width: 120 },
{ field: 'ShipCity', headerText: 'Ship City', width: 100 },
{ field: 'Employee.City', headerText: 'Employee City', width: 150 }
],
height: 315
});
grid.appendTo('#Grid');
`
```

Foreign key column in ##Platform_Name## Grid control

The Foreign key column in the Syncfusion Grid control allows you to display related data from a foreign key data source in a column within the grid. This feature is particularly useful when you have a column in the grid that represents a foreign key relationship with another data source.

To enable and integrate the foreign key column in the ##Platform_Name## Grid control, follow these steps:

1. Inject the ForeignKeyService in the Grid

```
ej.grids.Grid.Inject(ej.grids.ForeignKey);
`
```

2. Define the foreign key column in the grid using the following properties: * [dataSource](#): Specifies the foreign data source that contains the related data. * [foreignKeyField](#): Maps the column name in the grid to the field in the foreign data source that represents the foreign key relationship. * [foreignKeyValue](#): Specifies the field from the foreign data source that should be displayed in the grid as the related data.

```
{field: 'EmployeeID', headerText: 'Employee ID', width: 150, foreignKeyValue: 'FirstName',
foreignKeyField: 'EmployeeID', dataSource: employeeData}
`
```

The `foreignKeyField` property should match the name of the field in the foreign data source that represents the foreign key relationship, and the `foreignKeyValue` property should specify the field from the foreign data source that should be displayed in the grid as the related data.

Binding local data

The Syncfusion Grid control provides a convenient way to bind local data to a foreign key column. This allows you to display related data from a local data source within the grid. Here's an example of how to bind local data to a Foreign Key column in Syncfusion Grid:

In this example, **data** is the local data source for the Grid, and **employeeData** is the local data source for the foreign key column. The `field` property of the column is set to **EmployeeID** which represents the foreign key value in the **data**. The `foreignKeyValue` property is set to **FirstName** which represents the field name in the **employeeData** that you want to display in the foreign key column.

INDEX.JS

```
ej.grids.Grid.Inject(ej.grids.ForeignKey);
```

```

var grid = new ej.grids.Grid({
    dataSource: data,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign:
'Right', width: 100 },
        {field: 'EmployeeID', headerText: 'Employee Name', width:
120, foreignKeyValue: 'FirstName', dataSource: employeeData},
        { field: 'Freight', headerText: 'Freight', textAlign:
'Right', width: 80},
        { field: 'ShipCity', headerText: 'Ship City', width: 130 },
    ],
    height: 315
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <div id="container">
        <div id="Grid"></div>
    </div>

```

```

<script>
    var ele = document.getElementById('container');
    if (ele) {
        ele.style.visibility = "visible";
    }
</script>
<script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Binding remote data

The Foreign key column in Syncfusion Grid allows you to bind remote data for a foreign key column. You can assign the service data as an instance of **DataManager** to the **dataSource** property, and provide the endpoint **URL** as the data source URL.

This example demonstrates how to use the foreign key column with remote data binding using the [ODataV4Adaptor](#) in the grid:

INDEX.JS

```

var data = new ej.data.DataManager({
    url: 'https://services.odata.org/V4/Northwind/Northwind.svc/Orders/',
    adaptor: new ej.data.ODataV4Adaptor()
});
var employeeData = new ej.data.DataManager({
    url: 'https://services.odata.org/V4/Northwind/Northwind.svc/Employees/',
    adaptor: new ej.data.ODataV4Adaptor()
});
ej.grids.Grid.Inject(ej.grids.ForeignKey);
var grid = new ej.grids.Grid({
    dataSource: data,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign:
'Right', width: 100 },
        {field: 'EmployeeID', headerText: 'Employee Name', width:
120, foreignKeyValue: 'FirstName', dataSource: employeeData},
        { field: 'Freight', headerText: 'Freight', textAlign:
'Right', width: 80},
        { field: 'ShipCity', headerText: 'Ship City', width: 130 },
    ],
    height: 315
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
<body>
  <div id="container">
    <div id="Grid"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

* For remote data, the sorting and grouping is done based on [column.foreignKeyField](#) instead of [column.foreignKeyValue](#).

* If [column.foreignKeyField](#) is not defined, then the column uses [column.field](#).

Use edit template in foreignkey column

The Syncfusion Grid provides support for using an edit template in a foreign key column. By default, a dropdown control is used for editing foreign key column. However, you can render a different control for editing by using the [column.edit](#) property. Here's an example that demonstrates how to use an edit template in a foreign key column:

In this example, an [AutoComplete](#) control is rendered as the edit template for the “EmployeeID” foreign key column. The [dataSource](#) property of the [AutoComplete](#) control is set to the employees data, and the fields property is configured to display the “FirstName” field as the value.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.ForeignKey, ej.grids.Edit, ej.grids.Toolbar);
var autoCompleteObj;
var grid = new ej.grids.Grid({
  dataSource: data,
  toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
  editSettings: { allowEditing: true },
  columns: [
    { field: 'OrderID', headerText: 'Order ID', isPrimaryKey: true,
      textAlign: 'Right', width: 100 },
    {
      field: 'EmployeeID', headerText: 'Employee Name', width: 150,
      foreignKeyValue: 'FirstName', dataSource: employeeData,
      edit: {
        create: function () {
          // to create input element
          return ej.base.createElement('input');
        },
        read: function () {
          // return edited value to update data source
          var value = new ej.data.DataManager(employeeData).executeLocal(
            new ej.data.Query().where('FirstName', 'equal',
              autoCompleteObj.value)
          );
          return value.length && value[0]['EmployeeID']; // to convert
            foreign key value to local value.
        },
        destroy: function () {
          // to destroy the custom component.
          autoCompleteObj.destroy();
        },
        write: function (args) {
          autoCompleteObj = new ej.dropdowns.AutoComplete({
            dataSource: employeeData,
            fields: { value: args.column.foreignKeyValue },
            value: args.foreignKeyData[0][args.column.foreignKeyValue],
          });
          autoCompleteObj.appendTo(args.element);
        }
      },
    },
    { field: 'Freight', headerText: 'Freight', textAlign: 'Right', width: 80 },
    { field: 'ShipCity', headerText: 'Ship City', width: 130 },
  ],
  height: 270,
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Grid</title>

```

```

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container">
    <div id="Grid"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Customize filter UI in foreignkey column

The Syncfusion Grid allows you to customize the filtering user interface (UI) for foreign key columns by using the [column.filter](#) property. By default, a dropdown control is used for filtering foreign key columns. However, you can create your own custom filtering UI by specifying a template function for the [column.filter](#) property. Here's an example that demonstrates how to create a custom filtering UI in a foreign key column:

In this example, a [DropDownList](#) control is rendered as the filter UI for the “EmployeeID” foreign key column. The [dataSource](#) property of the DropDownList control is set to the employees data, and the

fields property is configured to display the **FirstName** field as the [text](#) and **EmployeeID** field as the [value](#). The **value** property is set to the current filter value of the column.

INDEX.JS

```
ej.grids.Grid.Inject(ej.grids.ForeignKey, ej.grids.Filter);
var dropInstance;
var grid = new ej.grids.Grid({
  dataSource: data,
  allowFiltering: true,
  filterSettings: { type: 'Menu' },
  columns: [
    {field: 'OrderID', headerText: 'Order ID', textAlign: 'Right', width:
100, },
    {field: 'EmployeeID', headerText: 'Employee Name', width:
150, foreignKeyValue: 'FirstName', dataSource: fEmployeeData,
    filter: {
      ui: {
        create: function (args) {
          var flValInput = new ej.base.createElement('input', {
            className: 'flm-input',
          });
          args.target.appendChild(flValInput);
          dropInstance = new ej.dropdowns.DropDownList({
            dataSource: new ej.data.DataManager(fEmployeeData),
            fields: { text: 'FirstName', value: 'FirstName' },
            placeholder: 'Select a value',
            popupHeight: '200px',
          });
          dropInstance.appendTo(flValInput);
        },
        write: function (args) {
          dropInstance.value = args.filteredValue;
        },
        read: function (args) {
          args.fltrObj.filterByColumn(
            args.column.field,
            args.operator,
            dropInstance.value
          );
        },
      },
    },
  ],
  { field: 'Freight', headerText: 'Freight', width: 100, textAlign:
'Right' },
  { field: 'ShipCity', headerText: 'Ship City', width: 180 },
],
  height: 315,
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container">
    <div id="Grid"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>
```

Use filter bar template in foreignkey column

You can use the filter bar template in a foreign key column in Grid by defining the [column.filterBarTemplate](#) property. This allows you to customize the filter bar for the foreign key column with a custom control or HTML template. Here's an example that demonstrates how to use a filter bar template in a foreign key column:

In this example, the “**EmployeeID**” column is a foreign key column, and the **filter** function is used as the filter bar template for this column. The **filter** function can be defined in your control code and should return the desired control or HTML template for the filter bar. The column header shows the custom filter bar template and you can select filter value by using the **DropDown** options.

INDEX.JS

```
ej.grids.Grid.Inject(ej.grids.ForeignKey, ej.grids.Filter);
var grid = new ej.grids.Grid({
  dataSource: data,
  allowFiltering: true,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100 },
    {
      field: 'EmployeeID', headerText: 'Employee Name', width: 120,
foreignKeyValue: 'FirstName', dataSource: fEmployeeData,
filterBarTemplate: {
      create: function() {
        return ej.base.createElement('input', { className: 'flm-
input' });
      },
      write: function(args) {
        fEmployeeData.splice(0, 0, { 'FirstName': 'All' }); //
for clear filtering
        var dropInstance = new ej.dropdowns.DropDownList({
          dataSource: new ej.data.DataManager(fEmployeeData),
          fields: { text: 'FirstName' },
          placeholder: 'Select a value',
          popupHeight: '200px',
          index: 0,
          change: function(e) {
            if (e.value !== 'All') {
              grid.filterByColumn('EmployeeID', 'equal',
e.value);
            }
            else {
              grid.removeFilteredColsByField('EmployeeID');
            }
          }
        });
        dropInstance.appendTo(args.element);
      }
    },
    { field: 'ShipCity', headerText: 'Ship City', width: 130 }
  ],
  height: 260
});
```

```
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container">
    <div id="Grid"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>
```

Perform aggregation in foreignkey column

By default, aggregations are not supported in a foreign key column in the Syncfusion Grid. However, you can achieve aggregation for a foreign key column by using [customAggregate](#).

To perform aggregation in a foreign key column, follow these steps:

1. Define a foreign key column in the Grid. 2. Implement a custom aggregate function to calculate the aggregation for the foreign key column. 3. Set the [customAggregate](#) property of the column to the custom aggregate function.

Here's an example that demonstrates how to perform aggregation in a foreign key column:

In the provided example, the `customAggregateFn` function is used to filter the data based on the **FirstName** field of the foreign key column, using the `getForeignData` internal function. The function then counts the occurrences of **Margaret**. The result is displayed in the grid's footer template using the `footerTemplate`.

INDEX.JS

```
ej.grids.Grid.Inject(ej.grids.ForeignKey, ej.grids.Aggregate);
var grid = new ej.grids.Grid({
  dataSource: data,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100, },
    { field: 'EmployeeID', headerText: 'Employee Name', width: 120,
foreignKeyValue: 'FirstName', dataSource: employeeData, },
    { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 80 },
    { field: 'ShipCity', headerText: 'Ship City', width: 130 },
  ],
  height: 280,
  aggregates: [
    {
      columns: [
        {
          type: 'Custom',
          customAggregate: customAggregateFn,
          field: 'EmployeeID',
          footerTemplate: 'Count of Margaret: ${Custom}',
        },
      ],
    },
  ],
});
grid.appendTo('#Grid');
function customAggregateFn(data1, column) {
  return data1.result.filter((count) => {
    return (
      ej.base.getValue(
        'FirstName',
        ej.grids.getForeignData(grid.getColumnByField(column.field),
count) [0]
      ) === 'Margaret'
    );
  }).length;
}
```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container">
    <div id="Grid"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Enable multiple foreign key columns

The Syncfusion Grid control supports the feature of enabling multiple foreign key columns with editing options. This allows users to display columns from foreign data sources in the Grid control.

In the following example, **Customer Name** and **Ship City** are foreign key columns that display the **ContactName** and **City** columns from foreign data.

INDEX.JS

```
ej.grids.Grid.Inject(ej.grids.Page, ej.grids.Edit, ej.grids.Toolbar,
ej.grids.ForeignKey);
var grid = new ej.grids.Grid({
    dataSource: orderDetails,
    allowPaging: true,
    toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
    editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
isPrimaryKey: true, width: 100 },
        { field: 'CustomerID', foreignKeyField: 'CustomerID',
foreignKeyValue: 'ContactName', dataSource: customerData, width: 170,
headerText: 'Customer Name', validationRules: { required: true } },
        { field: 'Freight', headerText: 'Freight', editType: 'numericedit',
width: 130, textAlign: 'Right', format: 'C2' },
        { field: 'EmployeeID', foreignKeyField: 'EmployeeID',
foreignKeyValue: 'City', dataSource: employeeData, width: 150, headerText:
'Ship City', validationRules: { required: true } },
        { field: 'ShipCountry', headerText: 'Ship Country', width: 150 }
    ],
    height: 270
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container">
    <div id="Grid"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Auto fit columns in ##Platform_Name## Grid control

The [autoFitColumns](#) method resizes the column to fit the widest cell's content without wrapping. You can autofit a specific column at initial rendering by invoking the [autoFitColumns](#) method in [dataBound](#) event.

To use the [autoFitColumns](#) method, inject the **Resize** module in the grid.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Resize);
var grid = new ej.grids.Grid({
  dataSource: data,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', width: 140 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 130 },
    { field: 'ShipName', headerText: 'Ship Name', width: 120 },
    { field: 'ShipCity', headerText: 'Ship City', width: 120 },
    { field: 'ShipAddress', headerText: 'Ship Address', width: 150 }
  ],
  dataBound: function() {
    grid.autoFitColumns(['ShipName', 'ShipAddress'])
  },
  height: 315
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>

```



```

<title>EJ2 Grid</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

You can autofit all the columns by invoking the [autoFitColumns](#) method without column names.

Column reorder in ##Platform_Name## Grid control

Reordering can be done by drag and drop of a particular column header from one index to another index within the grid. To enable reordering, set the [allowReordering](#) to true.

To use reordering, inject the [Reorder](#) module in the grid.

INDEX.JS

```
ej.grids.Grid.Inject(ej.grids.Reorder);
var grid = new ej.grids.Grid({
  dataSource: data,
  allowReordering: true,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
    { field: 'ShipCity', headerText: 'Ship City', width: 100 },
    { field: 'ShipName', headerText: 'Ship Name', width: 100 }
  ],
  height: 315
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <script id="rowtemplate" type="text/x-template">
    <tr>
      <td class="photo">
        
      </td>
      <td class="details">
        <table class="CardTable" cellpadding="3" cellspacing="2">
          <colgroup>
            <col width="50%">
            <col width="50%">
          </colgroup>
          <tbody>
            <tr>
              <td class="CardHeader">First Name </td>
              <td>{FirstName} </td>
            </tr>
            <tr>
              <td class="CardHeader">Last Name</td>
              <td>{LastName} </td>
            </tr>
            <tr>
              <td class="CardHeader">Title
              </td>
              <td>{Title}
              </td>
            </tr>
            <tr>
              <td class="CardHeader">Country
              </td>
              <td>{Country}
              </td>
            </tr>
          </tbody>
        </table>
      </td>
    </tr>
  </script>

  <div id="container">
    <div id="Grid"></div>
  </div>
</script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>

```

```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

You can disable reordering a particular column by setting the [columns.allowReordering](#) to **false**.

Reorder single column

Grid have option to reorder Columns either by Interaction or by using the [reorderColumns](#) method. In the below sample, **ShipCity** column is reordered to last column position by using the method.

INDEX.JS

```
ej.grids.Grid.Inject(ej.grids.Reorder);
var grid = new ej.grids.Grid({
  dataSource: data,
  allowReordering: true,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
    { field: 'ShipCity', headerText: 'Ship City', width: 100 },
    { field: 'ShipRegion', headerText: 'Ship Region', width: 100 },
    { field: 'ShipName', headerText: 'Ship Name', width: 100 }
  ],
  height: 280
});
grid.appendTo('#Grid');
var reorderSingleColsBtn = new ej.buttons.Button();
reorderSingleColsBtn.appendTo('#reorderSingleCol');
document.getElementById('reorderSingleCol').addEventListener('click',
function(){
  grid.reorderColumns('ShipCity', 'ShipName');
});
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
#reorderMultipleCols {
    text-transform: none;
}
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
<script id="rowtemplate" type="text/x-template">
<tr>
<td class="photo">

</td>
<td class="details">
<table class="CardTable" cellpadding="3" cellspacing="2">
<colgroup>
<col width="50%">
<col width="50%">
</colgroup>
<tbody>
<tr>
<td class="CardHeader">First Name </td>
<td>{FirstName} </td>
</tr>
<tr>
<td class="CardHeader">Last Name</td>
<td>{LastName} </td>
</tr>
<tr>
<td class="CardHeader">Title
</td>
<td>{Title}
</td>
</tr>
<tr>
<td class="CardHeader">Country
</td>
<td>{Country}
</td>
</tr>
</tbody>
</table>
</td>
</tr>

```

```

        </tbody>
    </table>
</td>
</tr>
</script>

<div id="container">
    <button id="reorderSingleCol">Reorder Ship City to Last</button>
    <div id="Grid"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Reorder multiple columns

User can reorder a single column at a time by Interaction. Sometimes we need to have reorder multiple columns at the same time, It can be achieved through programmatically by using [reorderColumns](#) method.

In the below sample, **Ship City** and **Ship Region** column is reordered to last column position.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Reorder);
var grid = new ej.grids.Grid({
    dataSource: data,
    allowReordering: true,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
        { field: 'ShipCity', headerText: 'Ship City', width: 100 },
        { field: 'ShipRegion', headerText: 'Ship Region', width: 100 },
        { field: 'ShipName', headerText: 'Ship Name', width: 100 }
    ],
    height: 280
});
grid.appendTo('#Grid');
var reorderMultipleCols = new ej.buttons.Button();
reorderMultipleCols.appendTo('#reorderMultipleCols');
document.getElementById('reorderMultipleCols').addEventListener('click',
function() {
    grid.reorderColumns(['ShipCity', 'ShipRegion'], 'ShipName');
});

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

```

```

<meta name="description" content="Typescript Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
#reorderMultipleCols {
    text-transform: none;
}
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
<script id="rowtemplate" type="text/x-template">
<tr>
<td class="photo">

</td>
<td class="details">
<table class="CardTable" cellpadding="3" cellspacing="2">
<colgroup>
<col width="50%">
<col width="50%">
</colgroup>
<tbody>
<tr>
<td class="CardHeader">First Name </td>
<td>{FirstName} </td>

```

```

        </tr>
        <tr>
            <td class="CardHeader">Last Name</td>
            <td>${LastName} </td>
        </tr>
        <tr>
            <td class="CardHeader">Title
            </td>
            <td>${Title}
            </td>
        </tr>
        <tr>
            <td class="CardHeader">Country
            </td>
            <td>${Country}
            </td>
        </tr>
    </tbody>
</table>
</td>
</tr>
</script>

<div id="container">
    <button id="reorderMultipleCols">Reorder Ship City and Ship Region
to Last</button>
    <div id="Grid"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Reorder events

During the reorder action, the grid component triggers the below three events.

1. The [columnDragStart](#) event triggers when column header element drag (move) starts.
2. The [columnDrag](#) event triggers when column header element is dragged (moved) continuously.
3. The [columnDrop](#) event triggers when a column header element is dropped on the target column.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Reorder);
var grid = new ej.grids.Grid({
    dataSource: data,
    allowReordering: true,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100 },

```



```

        { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
        { field: 'ShipCity', headerText: 'Ship City', width: 100 },
        { field: 'ShipName', headerText: 'Ship Name', width: 100 }
    ],
    columnDragStart: function() {
        alert('columnDragStart event is Triggered');
    },
    columnDrag: function() {
        alert('columnDrag event is Triggered');
    },
    columnDrop: function() {
        alert('columnDrop event is Triggered');
    },
    height: 315
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>

```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <script id="rowtemplate" type="text/x-template">
    <tr>
      <td class="photo">
        
      </td>
      <td class="details">
        <table class="CardTable" cellpadding="3" cellspacing="2">
          <colgroup>
            <col width="50%">
            <col width="50%">
          </colgroup>
          <tbody>
            <tr>
              <td class="CardHeader">First Name </td>
              <td>{FirstName} </td>
            </tr>
            <tr>
              <td class="CardHeader">Last Name</td>
              <td>{LastName} </td>
            </tr>
            <tr>
              <td class="CardHeader">Title
              </td>
              <td>{Title}
              </td>
            </tr>
            <tr>
              <td class="CardHeader">Country
              </td>
              <td>{Country}
              </td>
            </tr>
          </tbody>
        </table>
      </td>
    </tr>
  </script>

  <div id="container">
    <div id="Grid"></div>
  </div>
</script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Column resizing in ##Platform_Name## Grid control

Column width can be resized by clicking and dragging the right edge of the column header. While dragging, the width of the respective column will be resized immediately. Each column can be auto resized by double-clicking the right edge of the column header to fit the width of that column based on the widest cell content. To enable column resize, set the [allowResizing](#) property to true.

To use the column resize, inject **Resize** module in the grid.

INDEX.JS

```
ej.grids.Grid.Inject(ej.grids.Resize);
var grid = new ej.grids.Grid({
  dataSource: data,
  allowResizing: true,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width:150 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
    { field: 'Freight', headerText: 'Freight', width: 150 },
    { field: 'ShipCity', headerText: 'Ship City', width: 150 },
    { field: 'ShipName', headerText: 'Ship Name', width: 150 },
    { field: 'ShipAddress', headerText: 'Ship Address', width: 150 },
    { field: 'ShipCountry', headerText: 'Ship Country', width: 150 },
    { field: 'ShipPostalCode', headerText: 'Ship Postal code', width:
150 }
  ],
  height: 315
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <script id="rowtemplate" type="text/x-template">
    <tr>
      <td class="photo">
        
      </td>
      <td class="details">
        <table class="CardTable" cellpadding="3" cellspacing="2">
          <colgroup>
            <col width="50%">
            <col width="50%">
          </colgroup>
          <tbody>
            <tr>
              <td class="CardHeader">First Name </td>
              <td>{FirstName} </td>
            </tr>
            <tr>
              <td class="CardHeader">Last Name</td>
              <td>{LastName} </td>
            </tr>
            <tr>
              <td class="CardHeader">Title
              </td>
              <td>{Title}
              </td>
            </tr>
            <tr>
              <td class="CardHeader">Country
              </td>
              <td>{Country}
              </td>
            </tr>
          </tbody>
        </table>
      </td>
    </tr>
  </script>

  <div id="container">

```

```

        <div id="Grid"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

* You can disable resizing for a particular column by setting the [columns.allowResizing](#) to false.

* In RTL mode, you can click and drag the left edge of the header cell to resize the column.

Column resizing externally

To resize a column, set width to that particular column and then refresh the grid header by using the [refreshHeader\(\)](#) method. Please refer the below code

```

`ts
var grid = document.getElementById('Grid').ej2_instances[0]; //Grid Instance
var columns = grid.columns;
columns[0].width = 150;
grid.refreshHeader();
`

```

Min and max width

Column resize can be restricted between minimum and maximum width by defining the [columns->minWidth](#) and [columns->maxWidth](#).

In the following sample, minimum and maximum width are defined for **OrderID**, **Ship Name**, and **Ship Country** columns.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Resize);
var grid = new ej.grids.Grid({
    dataSource: data,
    allowResizing: true,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
minWidth: 100, width: 150, maxWidth: 300 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'Freight', headerText: 'Freight', width: 150 },
        { field: 'ShipCity', headerText: 'Ship City', width: 150 },
        { field: 'ShipName', headerText: 'Ship Name', minWidth: 120, width:
150, maxWidth: 250 },
        { field: 'ShipAddress', headerText: 'Ship Address', width: 150 },
        { field: 'ShipCountry', headerText: 'Ship Country', minWidth: 150,
width: 200, maxWidth: 350 },
        { field: 'ShipPostalCode', headerText: 'Ship Postal code', width:
150 }
    ],

```

```

        height: 315
    });
    grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <script id="rowtemplate" type="text/x-template">
        <tr>
            <td class="photo">
                
            </td>
            <td class="details">
                <table class="CardTable" cellpadding="3" cellspacing="2">
                    <colgroup>
                        <col width="50%">

```

```

        <col width="50%">
    </colgroup>
    <tbody>
        <tr>
            <td class="CardHeader">First Name </td>
            <td>${FirstName} </td>
        </tr>
        <tr>
            <td class="CardHeader">Last Name</td>
            <td>${LastName} </td>
        </tr>
        <tr>
            <td class="CardHeader">Title
            </td>
            <td>${Title}
            </td>
        </tr>
        <tr>
            <td class="CardHeader">Country
            </td>
            <td>${Country}
            </td>
        </tr>
    </tbody>
</table>
</td>
</tr>
</script>

<div id="container">
    <div id="Grid"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The `maxWidth` and `minWidth` properties will be considered only when the user resizes the column. When resizing the window, these properties will not be considered. This is because columns cannot be re-rendered when resizing the window.

Resize stacked column

Stacked columns can be resized by clicking and dragging the right edge of the stacked column header. While dragging, the width of the respective child columns will be resized at the same time. You can disable resize for any particular stacked column by setting `allowResizing` as **false** to its columns.

In this example, we have disabled resize for **Ship City** column.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Resize);
var grid = new ej.grids.Grid({

```

```

    dataSource: data,
    allowResizing: true,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100, minWidth: 10 },
        {
            headerText: 'Order Details', columns: [
                { field: 'OrderDate', headerText: 'Order Date',
textAlign: 'Right', width: 125, minWidth: 10, format: 'yMd' },
                { field: 'Freight', headerText: 'Freight($)', textAlign:
'Reight', width: 100, format: 'C2', minWidth: 10 },
            ]
        },
        {
            headerText: 'Ship Details', columns: [
                { field: 'ShipCity', headerText: 'Ship City', width:
100, minWidth: 10, allowResizing: false },
                { field: 'ShipCountry', headerText: 'Ship Country',
width: 120, minWidth: 10 },
            ]
        }
    ]
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <script id="rowtemplate" type="text/x-template">
    <tr>
      <td class="photo">
        
      </td>
      <td class="details">
        <table class="CardTable" cellpadding="3" cellspacing="2">
          <colgroup>
            <col width="50%">
            <col width="50%">
          </colgroup>
          <tbody>
            <tr>
              <td class="CardHeader">First Name </td>
              <td>{FirstName} </td>
            </tr>
            <tr>
              <td class="CardHeader">Last Name</td>
              <td>{LastName} </td>
            </tr>
            <tr>
              <td class="CardHeader">Title
              </td>
              <td>{Title}
              </td>
            </tr>
            <tr>
              <td class="CardHeader">Country
              </td>
              <td>{Country}
              </td>
            </tr>
          </tbody>
        </table>
      </td>
    </tr>
  </script>

  <div id="container">
    <div id="Grid"></div>
  </div>
</script>
var ele = document.getElementById('container');
if(ele) {

```

```

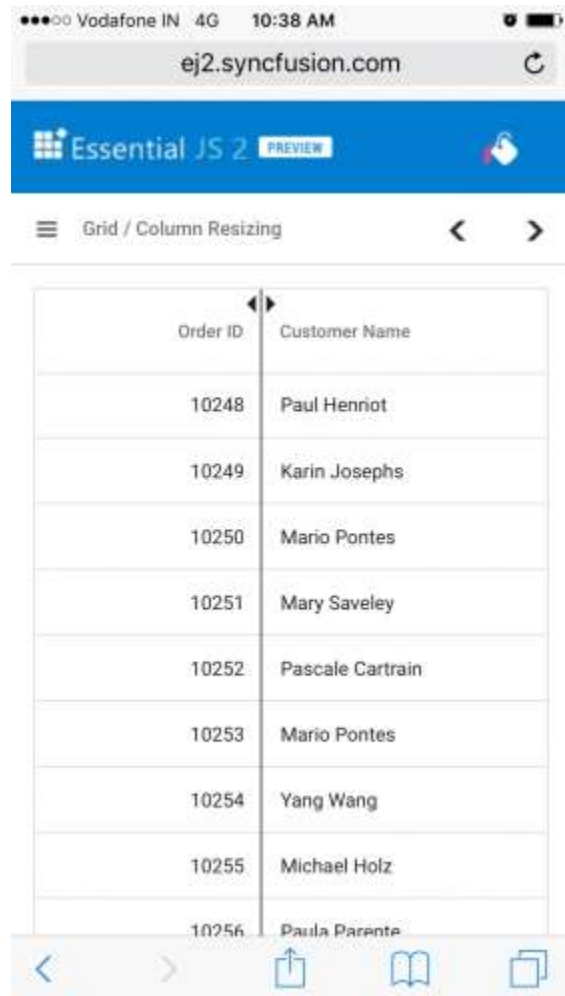
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Touch interaction

When the right edge of the header cell is tapped, a floating handler will be visible over the right border of the column. To resize the column, tap and drag the floating handler as needed. You can autoFit a column by using the Column menu of the grid.

The following screenshot represents the column resizing in touch device.



Resizing events

During the resizing action, the grid component triggers the below three events.

1. The [resizeStart](#) event triggers when column resize starts.
2. The [resizing](#) event triggers when column header element is dragged (moved) continuously..
3. The [resizeStop](#) event triggers when column resize ends.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Resize);
var grid = new ej.grids.Grid({
    dataSource: data,
    allowResizing: true,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width:150 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'Freight', headerText: 'Freight', width: 150 },
        { field: 'ShipCity', headerText: 'Ship City', width: 150 },
        { field: 'ShipName', headerText: 'Ship Name', width: 150 },
        { field: 'ShipAddress', headerText: 'Ship Address', width: 150 },
        { field: 'ShipCountry', headerText: 'Ship Country', width: 150 },
        { field: 'ShipPostalCode', headerText: 'Ship Postal code', width:
150 }
    ],
    resizeStart: function() {
        alert('ResizeStart event is Triggered');
    },
    resizing: function() {
        alert('Resizing event is Triggered');
    },
    resizeStop: function() {
        alert('ResizeStop event is Triggered');
    },
    height: 315
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <script id="rowtemplate" type="text/x-template">
    <tr>
      <td class="photo">
        
      </td>
      <td class="details">
        <table class="CardTable" cellpadding="3" cellspacing="2">
          <colgroup>
            <col width="50%">
            <col width="50%">
          </colgroup>
          <tbody>
            <tr>
              <td class="CardHeader">First Name </td>
              <td>{FirstName} </td>
            </tr>
            <tr>
              <td class="CardHeader">Last Name</td>
              <td>{LastName} </td>
            </tr>
            <tr>
              <td class="CardHeader">Title
              </td>
              <td>{Title}
              </td>
            </tr>
            <tr>
              <td class="CardHeader">Country
              </td>
              <td>{Country}
              </td>
            </tr>
          </tbody>
        </table>
      </td>
    </tr>
  </script>

  <div id="container">
    <div id="Grid"></div>
  </div>

```

```
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Column chooser in ##Platform_Name## Grid control

The column chooser feature in the Syncfusion ##Platform_Name## Grid control allows you to dynamically show or hide columns. This feature can be enabled by defining the [showColumnChooser](#) property as **true**.

To use the column chooser, inject the **ColumnChooser** module in the grid.

INDEX.JS

```
ej.grids.Grid.Inject(ej.grids.Toolbar, ej.grids.ColumnChooser);
var grid = new ej.grids.Grid({
    dataSource: data,
    showColumnChooser: true,
    toolbar: ['ColumnChooser'],
    columns: [
        { field: 'OrderID', headerText: 'Order ID', width: 120, textAlign:
'Right' },
        { field: 'OrderDate', headerText: 'Order Date', width: 130, format:
'yMd', textAlign: 'Right' },
        { field: 'Freight', headerText: 'Freight', width: 120, format: 'C2',
textAlign: 'Right' },
        { field: 'ShipCountry', headerText: 'Ship Country', width: 150 },
        { field: 'ShipCity', headerText: 'Ship City', visible: false, width:
150 }
    ],
    height: 235
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en">
<head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The column chooser dialog displays the header text of each column by default. If the header text is not defined for a column, the corresponding column field name is displayed instead.

Hide column in column chooser dialog

You can hide the column names in column chooser by defining the [columns->showInColumnChooser](#) as **false**. This feature is useful when working with a large number of columns or when you want to limit the number of columns that are available for selection in the column chooser dialog.

In this example, the `columns->showInColumnChooser` property is set to false for the **Order ID** column. As a result, the **Order ID** column will not be displayed in the column chooser dialog.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Toolbar, ej.grids.ColumnChooser);
var grid = new ej.grids.Grid({
    dataSource: data,
    showColumnChooser: true,
    toolbar: ['ColumnChooser'],
    columns: [
        { field: 'OrderID', headerText: 'Order ID', width: 120, textAlign:
'Right', showInColumnChooser: false },
        { field: 'OrderDate', headerText: 'Order Date', width: 130, format:
'yMd', textAlign: 'Right' },

```

```

        { field: 'Freight', headerText: 'Freight', width: 120, format: 'C2',
        textAlign: 'Right' },
        { field: 'ShipCountry', headerText: 'Ship Country', visible: false,
        width: 150 },
        { field: 'ShipCity', headerText: 'Ship City', visible: false, width:
        150 }
    ],
    height: 272
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  grids/styles/material.css" rel="stylesheet">
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>
  <div id="container">
    <div id="Grid"></div>
  </div>
  <script>
  var ele = document.getElementById('container');
  if(ele) {
    ele.style.visibility = "visible";
  }
  </script>

```

```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

The `columns->showInColumnChooser` property is applied to each column element individually. By setting it to false, you can hide specific columns from the column chooser dialog.

Open column chooser by external button

The Syncfusion `##Platform_Name##` Grid provides the flexibility to open the column chooser dialog on a web page using an external button. By default, the column chooser button is displayed in the right corner of the grid control, and clicking the button opens the column chooser dialog below it. However, you can programmatically open the column chooser dialog at specific **X** and **Y** axis positions by using the [openColumnChooser](#) method.

Here's an example of how to open the column chooser in the Grid using an external button:

INDEX.JS

```
ej.grids.Grid.Inject(ej.grids.ColumnChooser);
var grid = new ej.grids.Grid({
  dataSource: data,
  showColumnChooser: true,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', width: 120, textAlign:
    'Right', showInColumnChooser: false },
    { field: 'OrderDate', headerText: 'Order Date', width: 130, format:
    'yMd', textAlign: 'Right' },
    { field: 'Freight', headerText: 'Freight', width: 120, format: 'C2',
    textAlign: 'Right' },
    { field: 'ShipCountry', headerText: 'Ship Country', visible: false,
    width: 150 },
    { field: 'ShipCity', headerText: 'Ship City', visible: false, width:
    150 }
  ],
  height: 235
});
grid.appendTo('#Grid');
var showButton = new ej.buttons.Button({ cssClass: 'e-primary' }, '#show');
document.getElementById('show').onclick = function() {
  grid.columnChooserModule.openColumnChooser(100, 40); // give X and Y
axis
};
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en">
<head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <div id="container">
        <button id="show">Open Column Chooser</button>
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customize column chooser dialog size

The column chooser dialog in Syncfusion **##Platform_Name##** Grid comes with default size, but you can modify its height and width as per your specific needs using CSS styles.

To customize the column chooser dialog size, you can use the following CSS styles:

```

`css
.e-grid .e-dialog.e-ccdlg {
height: 500px;
width: 200px;
}
.e-grid .e-ccdlg .e-cc-contentdiv {
height: 200px;

```

width: 230px;

}

,

INDEX.JS

```
ej.grids.Grid.Inject(ej.grids.Toolbar, ej.grids.ColumnChooser);
var grid = new ej.grids.Grid({
  dataSource: data,
  showColumnChooser: true,
  toolbar: ['ColumnChooser'],
  columns: [
    { field: 'OrderID', headerText: 'Order ID', width: 120, textAlign:
'Right', showInColumnChooser: false },
    { field: 'OrderDate', headerText: 'Order Date', width: 130, format:
'yMd', textAlign: 'Right' },
    { field: 'Freight', headerText: 'Freight', width: 120, format: 'C2',
textAlign: 'Right' },
    { field: 'ShipCountry', headerText: 'Ship Country', visible: false,
width: 150 },
    { field: 'ShipCity', headerText: 'Ship City', visible: false, width:
150 }
  ],
  height: 235
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en">
<head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<style>
    .e-grid .e-dialog.e-ccdlg {
        height: 500px;
        width: 200px;
    }
    .e-grid .e-ccdlg .e-cc-contentdiv {
        height: 200px;
        width: 230px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Change default search operator of the column chooser

The column chooser dialog in the Syncfusion `##Platform_Name##` Grid provides a search box that allows you to search for column names. By default, the search functionality uses the "startswith" operator to match columns and display the results in the column chooser dialog. However, there might be cases where you need to change the default search operator to achieve more precise data matching.

To change the default search operator of the column chooser in Syncfusion Grid, you need to use the [operator](#) property of the `columnChooserSettings`.

Here's an example of how to change the default search operator of the column chooser to **contains** in the `##Platform_Name##` Grid:

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Toolbar, ej.grids.ColumnChooser);
var grid = new ej.grids.Grid({
    dataSource: data,
    showColumnChooser: true,
    toolbar: ['ColumnChooser'],
    columnChooserSettings: { operator: 'contains' },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', width: 120, textAlign:
'Right' },
        { field: 'OrderDate', headerText: 'Order Date', width: 120, format:
'yMd', textAlign: 'Right' },

```

```

        { field: 'Freight', headerText: 'Freight', width: 120, format: 'C2',
        textAlign: 'Right' },
        { field: 'ShipCountry', headerText: 'Ship Country', width: 130 },
        { field: 'ShipCity', headerText: 'Ship City', visible: false, width:
130 }
    ],
    height: 235
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en">
<head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>

```

```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Diacritics searching in column chooser

By default, the grid ignores diacritic characters when performing a search in the column chooser. However, in some cases, you may want to include diacritic characters in the search. To enable this behavior, you can set the [columnChooserSettings->ignoreAccent](#) property to **true**.

Here is an example that demonstrates the usage of the `ignoreAccent` property to include diacritic characters for searching in the column chooser:

INDEX.JS

```
ej.grids.Grid.Inject(ej.grids.Toolbar, ej.grids.ColumnChooser);
var grid = new ej.grids.Grid({
  dataSource: data,
  showColumnChooser: true,
  toolbar: ['ColumnChooser'],
  columnChooserSettings : { ignoreAccent: true },
  columns: [
    { field: 'OrderID^', headerText: 'Order ID^', width: 120, textAlign:
'Right' },
    { field: 'OrderDate', headerText: 'Order Date', width: 120, format:
'yMd', textAlign: 'Right' },
    { field: 'F^reight', headerText: 'F^reight', width: 120, format:
'C2', textAlign: 'Right' },
    { field: 'ShipCountry', headerText: 'Ship Country', width: 130 },
    { field: 'ShipCity', headerText: 'Ship City', visible: false, width:
130 }
  ],
  height: 272
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en">
<head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container">
    <div id="Grid"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Column menu in ##Platform_Name## Grid control

The column menu has options to integrate features like sorting, grouping, filtering, column chooser, and autofit. It will show a menu with the integrated feature when users click on multiple icon of the column. To enable column menu, you need to define the [showColumnMenu](#) property as true.

To use the column menu, inject the **ColumnMenu** module in the grid.

The default items are displayed in following table.

Item	Description
----- -----	
SortAscending	Sort the current column in ascending order.
SortDescending	Sort the current column in descending order.
Group	Group the current column.
Ungroup	Ungroup the current column.
AutoFit	Auto fit the current column.
AutoFitAll	Auto fit all columns.
ColumnChooser	Choose the column visibility.
Filter	Show the filter option as given in filterSettings.type

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Page, ej.grids.Group, ej.grids.Sort,
ej.grids.Filter, ej.grids.ColumnMenu);
var grid = new ej.grids.Grid({
  dataSource: data,
  allowGrouping: true,
  allowSorting: true,
  allowFiltering: true,
  filterSettings: { type: 'CheckBox' },
  allowPaging: true,
  groupSettings: { showGroupedColumn: true },
  showColumnMenu: true,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', width: 200, textAlign:
'Right',
      showInColumnChooser: false },
    { field: 'Freight', width: 150, format: 'C2', textAlign: 'Right',
editType: 'numericedit' },
    { field: 'ShipName', headerText: 'Ship Name', width: 300 },
    { field: 'ShipCountry', headerText: 'Ship Country', visible: false,
width: 200 },
    { field: 'ShipCity', headerText: 'Ship City', width: 200 }
  ]
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <script id="rowtemplate" type="text/x-template">
    <tr>
      <td class="photo">
        
      </td>
      <td class="details">
        <table class="CardTable" cellpadding="3" cellspacing="2">
          <colgroup>
            <col width="50%">
            <col width="50%">
          </colgroup>
          <tbody>
            <tr>
              <td class="CardHeader">First Name </td>
              <td>{FirstName} </td>
            </tr>
            <tr>
              <td class="CardHeader">Last Name</td>
              <td>{LastName} </td>
            </tr>
            <tr>
              <td class="CardHeader">Title
              </td>
              <td>{Title}
              </td>
            </tr>
            <tr>
              <td class="CardHeader">Country
              </td>
              <td>{Country}
              </td>
            </tr>
          </tbody>
        </table>
      </td>
    </tr>
  </script>

  <div id="container">
    <div id="Grid"></div>
  </div>
</script>
var ele = document.getElementById('container');
if(ele) {

```



```

    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

* You can disable column menu for a particular column by defining the [columns.showColumnMenu](#) as false.

* You can customize the default items by defining the [columnMenuItems](#) with required items.

Column menu events

During the resizing action, the grid component triggers the below two events.

1. The [columnMenuOpen](#) event triggers before the column menu opens.
2. The [columnMenuClick](#) event triggers when the user clicks the column menu of the grid.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Page, ej.grids.Group, ej.grids.Sort,
ej.grids.Filter, ej.grids.ColumnMenu);
var grid = new ej.grids.Grid({
  dataSource: data,
  allowGrouping: true,
  allowSorting: true,
  allowFiltering: true,
  filterSettings: { type: 'CheckBox' },
  allowPaging: true,
  groupSettings: { showGroupedColumn: true },
  showColumnMenu: true,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', width: 200, textAlign:
'Right',
      showInColumnChooser: false },
    { field: 'Freight', width: 150, format: 'C2', textAlign: 'Right',
editType: 'numericedit' },
    { field: 'ShipName', headerText: 'Ship Name', width: 300 },
    { field: 'ShipCountry', headerText: 'Ship Country', visible: false,
width: 200 },
    { field: 'ShipCity', headerText: 'Ship City', width: 200 }
  ],
  columnMenuOpen: function() {
    alert('columnMenuOpen event is Triggered');
  },
  columnMenuClick: function() {
    alert('columnMenuClick event is Triggered');
  }
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">

```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <script id="rowtemplate" type="text/x-template">
    <tr>
      <td class="photo">
        
      </td>
      <td class="details">
        <table class="CardTable" cellpadding="3" cellspacing="2">
          <colgroup>
            <col width="50%">
            <col width="50%">
          </colgroup>
          <tbody>
            <tr>
              <td class="CardHeader">First Name </td>
              <td>{FirstName} </td>
            </tr>
            <tr>
              <td class="CardHeader">Last Name</td>
              <td>{LastName} </td>
            </tr>
          </tbody>
        </table>
      </td>
    </tr>
  </script>

```

```

        </tr>
        <tr>
            <td class="CardHeader">Title
            </td>
            <td>${Title}
            </td>
        </tr>
        <tr>
            <td class="CardHeader">Country
            </td>
            <td>${Country}
            </td>
        </tr>
    </tbody>
</table>
</td>
</tr>
</script>

<div id="container">
    <div id="Grid"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Custom column menu item

Custom column menu items can be added by defining the [columnMenuItems](#) as collection of the [columnMenuItemModel](#). Actions for this customized items can be defined in the [columnMenuClick](#) event.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Page, ej.grids.Group, ej.grids.Sort,
ej.grids.Filter, ej.grids.ColumnMenu);
var grid = new ej.grids.Grid({
    dataSource: data,
    allowPaging: true,
    allowSorting: true,
    showColumnMenu: true,
    columnMenuItems:[{text:'Clear Sorting', id:'gridclearsorting'}],
    columnMenuClick: function(args){
        if(args.item.id === 'gridclearsorting'){
            grid.clearSorting();
        }
    },
    sortSettings:{
        columns:[{direction: "Ascending", field: "OrderID"}]
    },
    columns: [

```

```

        { field: 'OrderID', headerText: 'Order ID', width: 200, textAlign:
        'Right', showInColumnChooser: false },
        { field: 'Freight', width: 150, format: 'C2', textAlign: 'Right',
        editType: 'numericedit' },
        { field: 'ShipName', headerText: 'Ship Name', width: 300 },
        { field: 'ShipCountry', headerText: 'Ship Country', visible: false,
        width: 200 },
        { field: 'ShipCity', headerText: 'Ship City', width: 200 }
    ]
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <script id="rowtemplate" type="text/x-template">

```

```

<tr>
  <td class="photo">
    
  </td>
  <td class="details">
    <table class="CardTable" cellpadding="3" cellspacing="2">
      <colgroup>
        <col width="50%">
        <col width="50%">
      </colgroup>
      <tbody>
        <tr>
          <td class="CardHeader">First Name </td>
          <td>${FirstName} </td>
        </tr>
        <tr>
          <td class="CardHeader">Last Name</td>
          <td>${LastName} </td>
        </tr>
        <tr>
          <td class="CardHeader">Title
          </td>
          <td>${Title}
          </td>
        </tr>
        <tr>
          <td class="CardHeader">Country
          </td>
          <td>${Country}
          </td>
        </tr>
      </tbody>
    </table>
  </td>
</tr>
</script>

<div id="container">
  <div id="Grid"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customize menu items for particular columns

Sometimes, you have a scenario that to hide an item from column menu for particular columns. In that case, you need to define the [columnMenuOpenEventArgs.hide](#) as true in the [columnMenuOpen](#) event.

The following sample, **Filter** item was hidden in column menu when opens for the **OrderID** column.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Page, ej.grids.Group, ej.grids.Sort,
ej.grids.Filter, ej.grids.ColumnMenu);
var grid = new ej.grids.Grid({
    dataSource: data,
    allowPaging: true,
    showColumnMenu: true,
    filterSettings: {type: 'Menu'},
    allowFiltering: true,
    allowGrouping: true,
    allowSorting: true,
    columnMenuOpen: function (args) {
        for (var item of args.items) {
            if (item.text === 'Filter' && args.column.field === 'OrderID') {
                item.hide = true;
            } else {
                item.hide = false;
            }
        }
    },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', width: 200, textAlign:
'Right', showInColumnChooser: false },
        { field: 'Freight', width: 150, format: 'C2', textAlign: 'Right',
editType: 'numericedit' },
        { field: 'ShipName', headerText: 'Ship Name', width: 300 },
        { field: 'ShipCountry', headerText: 'Ship Country', visible: false,
width: 200 },
        { field: 'ShipCity', headerText: 'Ship City', width: 200 }
    ]
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <script id="rowtemplate" type="text/x-template">
    <tr>
      <td class="photo">
        
      </td>
      <td class="details">
        <table class="CardTable" cellpadding="3" cellspacing="2">
          <colgroup>
            <col width="50%">
            <col width="50%">
          </colgroup>
          <tbody>
            <tr>
              <td class="CardHeader">First Name </td>
              <td>{FirstName} </td>
            </tr>
            <tr>
              <td class="CardHeader">Last Name</td>
              <td>{LastName} </td>
            </tr>
            <tr>
              <td class="CardHeader">Title
              </td>
              <td>{Title}
              </td>
            </tr>
            <tr>
              <td class="CardHeader">Country
              </td>
              <td>{Country}
              </td>
            </tr>
          </tbody>
        </table>
      </td>
    </tr>
  </script>

```

```

    <div id="container">
      <div id="Grid"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customize the icon of column menu

You can customize the column menu icon by overriding the default grid class `.e-icons.e-columnmenu` with a custom property `content` as mentioned below.

```

.e-grid .e-columnheader .e-icons.e-columnmenu::before {
content: "\e941";
}

```

In the below sample, grid is rendered with a customized column menu icon.

INDEX.JS

```

var grid= new ej.grids.Grid({
  dataSource: data,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', width: 100 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
    { field: 'ShipName', headerText: 'Ship Name', width: 140 },
    { field: 'ShipCity', headerText: 'Ship City', width: 100 },
  ],
  showColumnMenu: true,
  height: 315
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

<style>
    .e-grid .e-columnheader .e-icons.e-columnmenu::before {
        content: "\e941";
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Column spanning in ##Platform_Name## Grid control

The column spanning feature in the Syncfusion Grid allows you to merge adjacent cells horizontally, creating a visually appealing and informative layout. By defining the [colSpan](#) attribute in the [queryCellInfo](#) event, you can easily span cells and customize the appearance of the grid.

In the following demo, Employee **Davolio** doing analysis from 9.00 AM to 10.00 AM, so that cells have spanned.

INDEX.JS

```

var grid = new ej.grids.Grid({
  dataSource: columnSpanData,
  queryCellInfo: QueryCellEvent,
  gridLines: 'Both',
  columns: [
    { field: 'EmployeeID', headerText: 'Employee ID', isPrimaryKey:
true, textAlign: 'Right', width: 150 },
    { field: 'EmployeeName', headerText: 'Employee Name', width: 200 },
    { field: '9:00', headerText: '9.00 AM', width: 120 },
    { field: '9:30', headerText: '9.30 AM', width: 120 },
    { field: '10:00', headerText: '10.00 AM', width: 120 },
    { field: '10:30', headerText: '10.30 AM', width: 120 },
    { field: '11:00', headerText: '11.00 AM', width: 120 },
    { field: '11:30', headerText: '11.30 AM', width: 120 },
    { field: '12:00', headerText: '12.00 PM', width: 120 },
    { field: '12:30', headerText: '12.30 PM', width: 120 },
    { field: '2:30', headerText: '2.30 PM', width: 120 },
    { field: '3:00', headerText: '3.00 PM', width: 120 },
    { field: '3:30', headerText: '3.30 PM', width: 120 },
    { field: '4:00', headerText: '4.00 PM', width: 120 },
    { field: '4:30', headerText: '4.30 PM', width: 120 },
    { field: '5:00', headerText: '5.00 PM', width: 120 }
  ],
  width: 'auto',
  height: 'auto',
  allowTextWrap: true
});
grid.appendTo('#Grid');
function QueryCellEvent(args) {
  var data = args.data;
  switch (data.EmployeeID) {
    case 10001:
      if (args.column.field === '9:00' || args.column.field === '2:30'
|| args.column.field === '4:30') {
        args.colSpan = 2;
      } else if (args.column.field === '11:00') {
        args.colSpan = 3;
      }
      break;
    case 10002:
      if (args.column.field === '9:30' || args.column.field === '2:30'
||
        args.column.field === '4:30') {
        args.colSpan = 3;
      } else if (args.column.field === '11:00') {
        args.colSpan = 4;
      }
      break;
    case 10003:
      if (args.column.field === '9:00' || args.column.field ===
'11:30') {
        args.colSpan = 3;
      } else if (args.column.field === '10:30' || args.column.field
=== '3:30' ||

```

```

        args.column.field === '4:30' || args.column.field ===
'2:30') {
            args.colSpan = 2;
        }
        break;
    case 10004:
        if (args.column.field === '9:00') {
            args.colSpan = 3;
        } else if (args.column.field === '11:00') {
            args.colSpan = 4;
        } else if (args.column.field === '4:00' || args.column.field ===
'2:30') {
            args.colSpan = 2;
        }
        break;
    case 10005:
        if (args.column.field === '9:00') {
            args.colSpan = 4;
        } else if (args.column.field === '11:30') {
            args.colSpan = 3;
        } else if (args.column.field === '3:30' || args.column.field ===
'4:30' || args.column.field === '2:30') {
            args.colSpan = 2;
        }
        break;
    case 10006:
        if (args.column.field === '9:00' || args.column.field === '4:30'
||
            args.column.field === '2:30' || args.column.field ===
'3:30') {
            args.colSpan = 2;
        } else if (args.column.field === '10:00' || args.column.field
=== '11:30') {
            args.colSpan = 3;
        }
        break;
    case 10007:
        if (args.column.field === '9:00' || args.column.field === '3:00'
|| args.column.field === '10:30') {
            args.colSpan = 2;
        } else if (args.column.field === '11:30' || args.column.field
=== '4:00') {
            args.colSpan = 3;
        }
        break;
    case 10008:
        if (args.column.field === '9:00' || args.column.field ===
'10:30' || args.column.field === '2:30') {
            args.colSpan = 3;
        } else if (args.column.field === '4:00') {
            args.colSpan = 2;
        }
        break;
    case 10009:
        if (args.column.field === '9:00' || args.column.field ===
'11:30') {
            args.colSpan = 3;
        }

```

```

    } else if (args.column.field === '4:30' || args.column.field ===
'2:30') {
        args.colSpan = 2;
    }
    break;
case 100010:
    if (args.column.field === '9:00' || args.column.field === '2:30'
||
        args.column.field === '4:00' || args.column.field ===
'11:30') {
        args.colSpan = 3;
    } else if (args.column.field === '10:30') {
        args.colSpan = 2;
    }
    break;
}
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en">
<head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <div id="container">

```

```

        <div id="Grid"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Change the border color while column spanning

You can change the border color for the spanned cells by the using [queryCellInfo](#) event. This event triggers before the cell element is appended to the Grid element.

INDEX.JS

```

var grid = new ej.grids.Grid({
    dataSource: columnSpanData,
    queryCellInfo: QueryCellEvent,
    gridLines: 'Both',
    columns: [
        { field: 'EmployeeID', headerText: 'Employee ID', isPrimaryKey: true,
        textAlign: 'Right', width: 150 },
        { field: 'EmployeeName', headerText: 'Employee Name', width: 200 },
        { field: '9:00', headerText: '9.00 AM', width: 120 },
        { field: '9:30', headerText: '9.30 AM', width: 120 },
        { field: '10:00', headerText: '10.00 AM', width: 120 },
        { field: '10:30', headerText: '10.30 AM', width: 120 },
        { field: '11:00', headerText: '11.00 AM', width: 120 },
        { field: '11:30', headerText: '11.30 AM', width: 120 },
        { field: '12:00', headerText: '12.00 PM', width: 120 },
        { field: '12:30', headerText: '12.30 PM', width: 120 },
        { field: '2:30', headerText: '2.30 PM', width: 120 },
        { field: '3:00', headerText: '3.00 PM', width: 120 },
        { field: '3:30', headerText: '3.30 PM', width: 120 },
        { field: '4:00', headerText: '4.00 PM', width: 120 },
        { field: '4:30', headerText: '4.30 PM', width: 120 },
        { field: '5:00', headerText: '5.00 PM', width: 120 }
    ],
    width: 'auto',
    height: 'auto',
    allowTextWrap: true
});
grid.appendTo('#Grid');
function QueryCellEvent(args) {
    var data = args.data;
    switch (data.EmployeeID) {
        case 10001:
            if (
                args.column.field === '9:00' ||
                args.column.field === '2:30' ||
                args.column.field === '4:30'
            ) {
                args.colSpan = 2;
            } else if (args.column.field === '11:00') {

```

```

        args.colSpan = 3;
    }
    break;
case 10002:
    if (
        args.column.field === '9:30' ||
        args.column.field === '2:30' ||
        args.column.field === '4:30'
    ) {
        args.colSpan = 3;
    } else if (args.column.field === '11:00') {
        args.colSpan = 4;
    }
    break;
case 10003:
    if (args.column.field === '9:00' || args.column.field === '11:30') {
        args.colSpan = 3;
    } else if (
        args.column.field === '10:30' ||
        args.column.field === '3:30' ||
        args.column.field === '4:30' ||
        args.column.field === '2:30'
    ) {
        args.colSpan = 2;
    }
    break;
case 10004:
    if (args.column.field === '9:00') {
        args.colSpan = 3;
    } else if (args.column.field === '11:00') {
        args.colSpan = 4;
    } else if (args.column.field === '4:00' || args.column.field ===
'2:30') {
        args.colSpan = 2;
    }
    break;
case 10005:
    if (args.column.field === '9:00') {
        args.colSpan = 4;
    } else if (args.column.field === '11:30') {
        args.colSpan = 3;
    } else if (
        args.column.field === '3:30' ||
        args.column.field === '4:30' ||
        args.column.field === '2:30'
    ) {
        args.colSpan = 2;
    }
    break;
case 10006:
    if (
        args.column.field === '9:00' ||
        args.column.field === '4:30' ||
        args.column.field === '2:30' ||
        args.column.field === '3:30'
    ) {
        args.colSpan = 2;
    }

```

```

    } else if (
        args.column.field === '10:00' ||
        args.column.field === '11:30'
    ) {
        args.colSpan = 3;
    }
    break;
case 10007:
    if (
        args.column.field === '9:00' ||
        args.column.field === '3:00' ||
        args.column.field === '10:30'
    ) {
        args.colSpan = 2;
    } else if (
        args.column.field === '11:30' ||
        args.column.field === '4:00'
    ) {
        args.colSpan = 3;
    }
    break;
case 10008:
    if (
        args.column.field === '9:00' ||
        args.column.field === '10:30' ||
        args.column.field === '2:30'
    ) {
        args.colSpan = 3;
    } else if (args.column.field === '4:00') {
        args.colSpan = 2;
    }
    break;
default:
    extendQueryCellEvent(args, data.EmployeeID);
}
if (args.colSpan > 1) {
    args.cell.style.border = '1px solid red';
}
}
function extendQueryCellEvent(args, value) {
    switch (value) {
        case 10009:
            if (args.column.field === '9:00' || args.column.field === '11:30') {
                args.colSpan = 3;
            } else if (args.column.field === '4:30' || args.column.field ===
'2:30') {
                args.colSpan = 2;
            }
            break;
        case 10010:
            if (
                args.column.field === '9:00' ||
                args.column.field === '2:30' ||
                args.column.field === '4:00' ||
                args.column.field === '11:30'
            ) {
                args.colSpan = 3;
            }
    }
}

```

```

    } else if (args.column.field === '10:30') {
        args.colSpan = 2;
    }
    break;
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en">
<head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <div id="container">
        <div id="Grid"></div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```


Responsive columns in ##Platform_Name## Grid control

You can toggle column visibility based on media queries which are defined at the [hideAtMedia](#).

The [hideAtMedia](#) accepts valid [Media Queries](#). In the below sample, for **OrderID** column, [hideAtMedia](#) property value is set as **(min-width: 700px)** so that **OrderID** column will get hidden when the browser screen width is less than 700px.

INDEX.JS

```
var grid = new ej.grids.Grid({
  dataSource: data,
  columns: [
    {
      field: 'OrderID', headerText: 'Order ID', width: 120, textAlign:
      'Right',
      hideAtMedia: '(min-width: 700px)'
    }, // column hides when browser screen width less than 700px;
    {
      field: 'CustomerID', headerText: 'Customer ID', width: 150,
      hideAtMedia: '(max-width: 500px)'
    }, // column shows when browser screen width less than or equal to
    500px;
    {
      field: 'OrderDate', headerText: 'Order Date', width: 130,
      format: 'yMd',
      textAlign: 'Right', hideAtMedia: '(min-width: 500px)'
    }, // column hides when browser screen size less than 500px;
    { field: 'Freight', width: 120, format: 'C2', textAlign: 'Right' }
  ], // it always shown
  height: 315
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  dropdowns/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Row

Row in ##Platform_Name## Grid control

The row represents record details fetched from data source.

Row customization

Using event

You can customize the appearance of a row by using the [rowDataBound](#) event. The [rowDataBound](#) event triggers for every row. In the event handler, you can get the

[RowDataBoundEventArgs](#) that contains details of the row.

INDEX.JS

```

var grid = new ej.grids.Grid({
    dataSource: data,
    enableHover: false,
    allowSelection: false,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100 },

```

```

        { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
        { field: 'ShipCity', headerText: 'Ship City', width: 100 },
        { field: 'Freight', headerText: 'Freight', width: 100, format: 'C2'
    },
        { field: 'ShipName', headerText: 'Ship Name', width: 100 }
    ],
    rowDataBound: rowBound,
    height: 280
});
grid.appendTo('#Grid');
function rowBound(args) {
    if (args.data['Freight'] < 30) {
        args.row.classList.add('below-30');
    } else if (args.data['Freight'] < 80) {
        args.row.classList.add('below-80');
    } else {
        args.row.classList.add('above-80');
    }
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Using CSS customize alternate rows

You can change the grid's alternative rows' background color by overriding the **.e-altrow** class.

```

.e-grid .e-altrow {
background-color: #fafafa;
}

```

Please refer to the following example.

INDEX.JS

```

var grid = new ej.grids.Grid({
    dataSource: data.slice(0, 8),
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
        { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C' },
        { field: 'OrderDate', headerText: 'Order Date', width: 140, format:
'yMd' }
    ]
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
<title>EJ2 Grid</title>
<meta charset="utf-8">

```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Grid Control">
<meta name="author" content="Syncfusion">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Using CSS customize selected row

The background color of the selected row can be changed by overriding the following CSS style.

,

```
.e-grid td.e-active {
background-color: #f9920b;
}
,
```

This is demonstrated in the following sample:

INDEX.JS

```
var grid = new ej.grids.Grid({
  dataSource: data.slice(0, 8),
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
    { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
    { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C' },
    { field: 'OrderDate', headerText: 'Order Date', width: 140, format:
'yMd' }
  ]
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
```

```

<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Adding a new row programmatically

The Grid can add a new row between the existing rows using the [addRecord](#) method of the Grid.

This is demonstrated in the following sample:

INDEX.JS

```

var grid = new ej.grids.Grid({
    dataSource: data,
    editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
        { field: 'ShipCity', headerText: 'Ship City', width: 100 },
        { field: 'Freight', headerText: 'Freight', width: 100, format: 'C2'
},
        { field: 'ShipName', headerText: 'Ship Name', width: 100 }
    ]
});
grid.appendTo('#Grid');
document.getElementById('addrow').onclick = function () {
    grid.addRecord(
        { OrderID: 3232, CustomerID: 'ALKIT', ShipCity: 'London', Freight:
40, ShipName: 'Que Delícia'}, 2);
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>

```

```

<title>EJ2 Grid</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="addrow">Add Row</button>
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

When working with remote data, it is impossible to add a new row between the existing rows.

How to get the row information when hovering over the cell

It is possible to get the row information when hovering over the specific cell. This can be achieved by using the [rowDataBound](#) event and [getRowInfo](#) method of the Grid.

In the following sample, the `mouseover` event is bound to a grid row in the `rowDataBound` event, and when hovering over the specific cell, using the `getRowInfo` method, row information will be retrieved and displayed in the console.

INDEX.JS

```
var grid = new ej.grids.Grid({
  dataSource: data,
  rowDataBound: rowDataBound,
  editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true },
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
    { field: 'ShipCity', headerText: 'Ship City', width: 100 },
    { field: 'Freight', headerText: 'Freight', width: 100, format: 'C2'
},
    { field: 'ShipName', headerText: 'Ship Name', width: 100 }
  ]
});
grid.appendTo('#Grid');
function rowDataBound(args) {
  args.row.addEventListener('mouseover', function(e) {
    console.log(grid.getRowInfo(e.target))
  })
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [How to maintain selected rows after adding new record](#)
- [How to select the specific record in the grid using its primary key value](#)
- [How to achieve drag and drop the rows in Grid with custom data binding](#)
- [How to get selected records on custom toolbar click](#)

Row height in ##Platform_Name## Grid control

You can customize the row height of grid rows through the [rowHeight](#) property. The **rowHeight** property

is used to change the row height of entire grid rows.

In the below example, the **rowHeight** is set as '60px'.

INDEX.JS

```
var grid = new ej.grids.Grid({
  dataSource: data.slice(0, 8),
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
    { field: 'CustomerID', width: 140, headerText: 'Customer ID',
type: 'string' },
    { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C' },
    { field: 'OrderDate', headerText: 'Order Date', width: 140,
format: 'yMd' }
  ],
  rowHeight: 60
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customize row height for particular row

Grid row height for particular row can be customized using the [rowDataBound](#) event by setting the **rowHeight** in arguments for each row based on the requirement.

In the below example, the row height for the row with OrderID as '10249' is set as '90px' using the **rowDataBound** event.

INDEX.JS

```

var gridData = new ej.data.DataManager(data).executeLocal(new
ej.data.Query().take(8));
var grid = new ej.grids.Grid({
    dataSource: gridData,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
        { field: 'CustomerID', width: 140, headerText: 'Customer ID',
type: 'string' },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C' },
        { field: 'OrderDate', headerText: 'Order Date', width: 140,
format: 'yMd' }
    ],
    rowDataBound: rowDataBound
});
function rowDataBound(args) {
    if((args.data).OrderID === 10249){
        args.rowHeight = 90;
    }
}
grid.appendTo('#Grid');

```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Grid"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

* In virtual scrolling mode, it is not applicable to set the **rowHeight** using the **rowDataBound** event.

Row template in ##Platform_Name## Grid control

The row template feature in Grid allows you to customize the appearance and layout of rows in the grid. This feature is useful when you want to display custom content, such as images, buttons, or other controls, within the rows.

To enable the row template feature, you need to set the [rowTemplate](#) property of the Grid control. This property accepts a custom HTML template that defines the layout for each row.

In the following example, Employee Information with Employee Photo is presented in the first column and employee details like Name, Address, etc., are presented in the second column.

INDEX.JS

```
var grid = new ej.grids.Grid({
  dataSource: employeeData,
  rowTemplate: '#rowtemplate',
  columns: [
    { headerText: 'Employee Image', width: 150, textAlign: 'Center',
      field: 'OrderID' },
    { headerText: 'Employee Details', width: 300, field: 'EmployeeID' }
  ],
  height: 315
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en">
<head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js" type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type="text/javascript"></script>
</head>
<body>
  <script id="rowtemplate" type="text/x-template">
    <tr>
      <td class="photo">
        
      </td>
      <td class="details">
        <table class="CardTable" cellpadding="3" cellspacing="2">
          <colgroup>
            <col width="50%">
            <col width="50%">
          </colgroup>
          <tbody>
            <tr>
              <td class="CardHeader">First Name </td>
              <td>{FirstName} </td>
            </tr>
            <tr>
              <td class="CardHeader">Last Name</td>
              <td>{LastName} </td>
            </tr>
            <tr>
              <td class="CardHeader">Title
              </td>
              <td>{Title}
              </td>
            </tr>
          </tbody>
        </table>
      </td>
    </tr>
  </script>

  <div id="container">
    <div id="Grid"></div>
  </div>
</script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Row template with formatting

The row template feature in Syncfusion Grid allows you to customize the layout of rows in the grid. This is useful when you want to display images, buttons, or other custom content within the rows of a grid.

By default, Syncfusion Grid provides the [columns->format](#) property to format the values displayed in each column. However, when using the [rowtemplate](#), the [columns->format](#) property cannot be directly applied to format the values inside the template.

To format the values within the row template, you can define a global function that handles the formatting logic. This function can be invoked inside the template to format the corresponding values.

Here is an example of how to define a global formatting function for a date column and use it inside a [rowTemplate](#):

INDEX.JS

```
var intl = new ej.base.Internationalization();
var dFormatter = intl.getDateFormat({ skeleton: 'yMd', type: 'date' });
(window).formatDate = (date) => dFormatter(date);
var grid = new ej.grids.Grid({
  dataSource: employeeData,
  rowTemplate: '#rowtemplate',
  columns: [
    { headerText: 'Employee Image', width: 150, textAlign: 'Center',
field: 'OrderID' },
    { headerText: 'Employee Details', width: 300, field: 'EmployeeID' }
  ],
  height: 315
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en">
<head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <script id="rowtemplate" type="text/x-template">
    <tr>
      <td class="photo">
        
      </td>
      <td class="details">
        <table class="CardTable" cellpadding="3" cellspacing="2">
          <colgroup>
            <col width="50%">
            <col width="50%">
          </colgroup>
          <tbody>
            <tr>
              <td class="CardHeader">First Name </td>
              <td>{FirstName} </td>
            </tr>
            <tr>
              <td class="CardHeader">Last Name</td>
              <td>{LastName} </td>
            </tr>
            <tr>
              <td class="CardHeader">Title
              </td>
              <td>{Title}
              </td>
            </tr>
            <tr>
              <td class="CardHeader">Birth Date
              </td>
              <td>{formatDate(data.BirthDate)}
              </td>
            </tr>
            <tr>
              <td class="CardHeader">Hire Date
              </td>
              <td>{formatDate(data.HireDate)}
              </td>
            </tr>
          </tbody>
        </table>
      </td>
    </tr>
  </script>
  <div id="container">

```

```

        <div id="Grid"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

When using the `rowTemplate` feature in Syncfusion Grid, keep in mind that any formatting applied to columns using the `columns->format` property will not work inside the template.

Render syncfusion control in row template

The Grid allows you to render custom Syncfusion controls within the rows of the grid. This feature is helpful as it enables you to display interactive Syncfusion controls instead of field values in the grid.

To enable a Syncfusion control in a row template, you need to set the `rowTemplate` property of the Grid control. This property accepts a custom HTML template that defines the layout for each row.

Here is an example that demonstrates rendering Syncfusion controls within a row template :

INDEX.JS

```

var grid = new ej.grids.Grid({
    dataSource: data,
    rowTemplate: '#rowtemplate',
    columns: [
        { field: 'OrderID', headerText: 'Order ID', width: 120 },
        { field: 'Quantity', headerText: 'Quantity', width: 170 },
        { field: 'ShipAddress', headerText: 'ShipAddress', width: 170 },
        { field: 'OrderDate', headerText: 'Order Date', width: 120 },
        { field: 'OrderStatus', headerText: 'Order Status', width: 120 },
    ],
    dataBound: () => {
        var gridInstance = document.getElementById('Grid').ej2_instances[0];
        var chipList =
gridInstance.getContentTable().querySelectorAll('.chipList');
        for (var i = 0; i < chipList.length; i++) {
            var chipValue = chipList[i].innerText;
            new ej.buttons.ChipList({ chips: [chipValue] }, chipList[i]);
        }
        var NumericList =
gridInstance.getContentTable().querySelectorAll('.numeric');
        for (var i = 0; i < NumericList.length; i++) {
            var numberValue = new ej.inputs.NumericTextBox({});
            numberValue.appendTo(NumericList[i]);
        }
        var dateList =
gridInstance.getContentTable().querySelectorAll('.date-input');
        for (var i = 0; i < dateList.length; i++) {
            var dateInput = dateList[i];
            var dateValue = dateInput.value;
            var datepicker = new ej.calendars.DatePicker({
                value: new Date(dateValue),
            });

```

```

    });
    datepicker.appendTo(dateInput);
}
var dropdownList =
gridInstance.getContentTable().querySelectorAll('.dropdownlist-input');
for (var i = 0; i < dropdownList.length; i++) {
    var dropdownInputValue = dropdownList[i];
    var dropData = ['Processing', 'Order Placed', 'Delivered'];
    var dropdown = new ej.dropdowns.DropDownList({
        dataSource: dropData,
        value: dropdownList[i].value,
        popupHeight: 150,
        popupWidth: 150
    });
    dropdown.appendTo(dropdownInputValue);
}
},
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>

```

```

</head>
<body>
  <script id="rowtemplate" type="text/x-template">
    <tr class="rows">
      <td class="chipList"> ${OrderID}</td>
      <td><input class="numeric" type="text" value=${Quantity}></td>
      <td>${ShipAddress} </td>
      <td><input class="date-input" value="${OrderDate}"></td>
      <td><input class="dropdownlist-input" type="text"
value="${OrderStatus}"></td>
    </tr>
  </script>
  <div id="container">
    <div id="Grid"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Limitations

Row template feature is not compatible with all the features which are available in the grid, and it has limited features support. The features that are incompatible with the row template feature are listed below.

- Filtering
- Paging
- Sorting
- Searching
- Rtl
- Export
- Context Menu
- State Persistence
- Selection
- Grouping
- Editing
- Frozen rows & columns
- Virtual & Infinite scrolling
- Column chooser
- Column menu
- Detail Row
- Foreignkey column
- Resizing
- Reordering
- Aggregates
- Clipboard

- Adaptive view

Detail template in ##Platform_Name## Grid control

The detail template in the Grid control allows you to display additional information about a specific row in the grid by expanding or collapsing detail content. This feature is useful when you need to show additional data or custom content that is specific to each row in the grid. You can use the [detailTemplate](#) property to define an HTML template for the detail row. This template can include any HTML element or ##Platform_Name## control that you want to display as detail content.

Here's an example of using the `detailTemplate` property in the grid control:

INDEX.JS

```
ej.grids.Grid.Inject(ej.grids.DetailRow);
var grid = new ej.grids.Grid({
  dataSource: employeeData,
  detailTemplate: '#detailtemplate',
  columns: [
    { field: 'FirstName', headerText: 'First Name', width: 140 },
    { field: 'LastName', headerText: 'Last Name', width: 140 },
    { field: 'Title', headerText: 'Title', width: 150 },
    { field: 'Country', headerText: 'Country', width: 150 }
  ],
  height: 315
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-grids/styles/material.css" rel="stylesheet">
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js" type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type="text/javascript"></script>
</head>
<body>
<script id="detailtemplate" type="text/x-template">
<table class="detailtable" width="100%" >
<colgroup>
<col width="35%">
<col width="35%">
<col width="30%">
</colgroup>
<tbody>
<tr>
<td rowspan="4" style="text-align: center;">

</td>
<td>
<span style="font-weight: 500;">First Name: </span>
{FirstName}
</td>
<td>
<span style="font-weight: 500;">Postal Code: </span>
{PostalCode}
</td>
</tr>
<tr>
<td>
<span style="font-weight: 500;">Last Name: </span>
{LastName}
</td>
<td>
<span style="font-weight: 500;">City: </span> {City}
</td>
</tr>
<tr>
<td>
<span style="font-weight: 500;">Title: </span> {Title}
</td>
<td>
<span style="font-weight: 500;">Phone: </span>
{HomePhone}
</td>
</tr>
<tr>
<td>
<span style="font-weight: 500;">City: </span> {City}
</td>
<td>

```

```

        <span style="font-weight: 500;">Country: </span>
    </td>
</tr>
</tbody>
</table>
</script>
<div id="container">
    <div id="Grid"></div>
</div>
<script>
    var ele = document.getElementById('container');
    if (ele) {
        ele.style.visibility = "visible";
    }
</script>
<script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Rendering custom control

The Grid control provides a powerful feature that allows you to render custom controls inside the detail row. This feature is helpful when you need to add additional information or functionality for a specific row in the grid.

To render a custom control inside the detail row, you need to define a template using the [detailTemplate](#) property and handle the [detailDataBound](#) event. This template can include any HTML element or `##Platform_Name##` control that you want to display as the detail content.

The `detailDataBound` event is an event that is triggered after a detail row is bound to data. This event provides an object of type [DetailDataBoundEventArgs](#) as a parameter.

For example, to render grid inside the detail row, place an HTML div element as the `detailTemplate` and render the DIV element as grid control in the `detailDataBound` event.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.DetailRow);
var grid = new ej.grids.Grid({
    dataSource: employeeData,
    detailTemplate: '#detailtemplate',
    columns: [
        { field: 'FirstName', headerText: 'First Name', width: 140 },
        { field: 'LastName', headerText: 'Last Name', width: 140 },
        { field: 'Title', headerText: 'Title', width: 150 },
        { field: 'Country', headerText: 'Country', width: 150 }
    ],
    detailDataBound: function(e){
        var detail = new ej.grids.Grid({
            dataSource: data.filter(function(item){ return
item['EmployeeID'] === e.data['EmployeeID'];}),
            columns: [
                { field: 'OrderID', headerText: 'Order ID', width: 110 },
                { field: 'CustomerID', headerText: 'Customer Name', width:
140 },

```

```

        { field: 'ShipCountry', headerText: 'Ship Country', width:
150 }
    ]
    });
    detail.appendTo(e.detailElement.querySelector('.custom-grid'));
}
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en">
<head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <script id="detailtemplate" type="text/x-template">
        <div class='custom-grid'></div>
    </script>
    <div id="container">
        <div id="Grid"></div>
    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";

```



```

}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Expand by external button

The Grid provides a feature that allows users to expand the detail row of a grid using an external button. By default, detail rows render in a collapsed state, but this feature enables users to view additional details associated with a particular row.

To achieve expanding the detail row of a grid using an external button, you need to invoke the [expand](#) method provided by the **detailRowModule** object of the Syncfusion Grid library. This method will expand the detail row of a specific grid row.

Here is an example of how to use the **expand** method to expand a detail row:

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.DetailRow);
var grid = new ej.grids.Grid({
  dataSource: employeeData,
  detailTemplate: '#detailtemplate',
  columns: [
    { field: 'FirstName', headerText: 'First Name', width: 140 },
    { field: 'LastName', headerText: 'Last Name', width: 140 },
    { field: 'Title', headerText: 'Title', width: 150 },
    { field: 'Country', headerText: 'Country', width: 150 }
  ],
  height: 260
});
grid.appendTo('#Grid');
var expandButton = new ej.buttons.Button();
expandButton.appendTo('#expand');
var textbox = new ej.inputs.TextBox({
  placeholder: 'Enter the Row Index',
  floatLabelType: 'Auto',
  width: 250
});
textbox.appendTo('#rowindex');
document.getElementById('expand').addEventListener('click', function(args){
  grid.detailRowModule.expand(textbox.value);
});

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en">
<head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
<script id="detailtemplate" type="text/x-template">
<table class="detailtable" width="100%" >
<colgroup>
<col width="35%">
<col width="35%">
<col width="30%">
</colgroup>
<tbody>
<tr>
<td rowspan="4" style="text-align: center;">

</td>
<td>
<span style="font-weight: 500;">First Name: </span>
${FirstName}
</td>
<td>
<span style="font-weight: 500;">Postal Code: </span>
${PostalCode}
</td>
</tr>
<tr>
<td>
<span style="font-weight: 500;">Last Name: </span>
${LastName}
</td>
<td>
<span style="font-weight: 500;">City: </span> ${City}
</td>
</tr>
<tr>
<td>
<span style="font-weight: 500;">City: </span> ${City}
</td>
<td>
<span style="font-weight: 500;">City: </span> ${City}
</td>
</tr>
<tr>
<td>
<span style="font-weight: 500;">City: </span> ${City}
</td>
<td>
<span style="font-weight: 500;">City: </span> ${City}
</td>
</tr>
</tbody>
</table>
</script>

```

```

                <td>
                    <span style="font-weight: 500;">Title: </span> ${Title}
                </td>
                <td>
                    <span style="font-weight: 500;">Phone: </span>
${HomePhone}
                </td>
            </tr>
            <tr>
                <td>
                    <span style="font-weight: 500;">City: </span> ${City}
                </td>
                <td>
                    <span style="font-weight: 500;">Country: </span>
${Country}
                </td>
            </tr>
        </tbody>
    </table>
</script>
    <div id="container">
        <div class="e-float-input" style="width: 250px; display: inline-
block ;padding: 0px 30px 0px 0px">
            <input id="rowindex" type="text" />
            <span class="e-float-line"></span>
        </div>
        <button id="expand">Expand</button>
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customize detail template icon

The detail template icon in the Syncfusion Grid is used to expand or collapse the detail content of a row. By default, the icon represents a right arrow for the collapsed state and a down arrow for the expanded state. If you want to customize this icon, you can achieve it by overriding the following CSS styles:

```

`css
.e-grid .e-icon-grightarrow::before {
content: '\e300';
}
.e-grid .e-icon-gdownarrow::before {
content: '\e304';
}

```

Here is an example of how to customize the detail template icon:

INDEX.JS

```
ej.grids.Grid.Inject(ej.grids.DetailRow);
var grid = new ej.grids.Grid({
  dataSource: employeeData,
  detailTemplate: '#detailtemplate',
  columns: [
    { field: 'FirstName', headerText: 'First Name', width: 140 },
    { field: 'LastName', headerText: 'Last Name', width: 140 },
    { field: 'Title', headerText: 'Title', width: 150 },
    { field: 'Country', headerText: 'Country', width: 150 }
  ],
  height: 315
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en">
<head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <style>
    .e-grid .e-icon-grightarrow::before {
      content: '\e300';
    }
  </style>
```

```

        .e-grid .e-icon-gdownarrow::before {
            content: '\e304';
        }
    </style>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
    </head>
    <body>
        <script id="detailtemplate" type="text/x-template">
            <table class="detailtable" width="100%" >
                <colgroup>
                    <col width="35%">
                    <col width="35%">
                    <col width="30%">
                </colgroup>
                <tbody>
                    <tr>
                        <td rowspan="4" style="text-align: center;">
                            
                        </td>
                        <td>
                            <span style="font-weight: 500;">First Name: </span>
                            {FirstName}
                        </td>
                        <td>
                            <span style="font-weight: 500;">Postal Code: </span>
                            {PostalCode}
                        </td>
                    </tr>
                    <tr>
                        <td>
                            <span style="font-weight: 500;">Last Name: </span>
                            {LastName}
                        </td>
                        <td>
                            <span style="font-weight: 500;">City: </span> {City}
                        </td>
                    </tr>
                    <tr>
                        <td>
                            <span style="font-weight: 500;">Title: </span> {Title}
                        </td>
                        <td>
                            <span style="font-weight: 500;">Phone: </span>
                            {HomePhone}
                        </td>
                    </tr>
                    <tr>
                        <td>
                            <span style="font-weight: 500;">City: </span> {City}
                        </td>
                        <td>
                            <span style="font-weight: 500;">Country: </span>
                            {Country}
                        </td>
                    </tr>
                </tbody>
            </table>
        </script>
    </body>
    </html>

```

```

        </tbody>
    </table>
</script>
    <div id="container">
        <div id="Grid"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Limitations

Detail template is not supported with the following features:

- Frozen rows and columns
- Immutable mode
- Infinite scrolling
- Virtual scrolling
- Print
- Row template
- Row spanning
- Column spanning
- Lazy load grouping
- State persistence

Row drag and drop in ##Platform_Name## Grid control

The Syncfusion ##Platform_Name## Grid control provides built-in support for row drag and drop functionality. This feature allows you to easily rearrange rows within the grid by dragging and dropping them to new positions. Additionally, you can also drag and drop rows from one grid to another grid, as well as drag and drop rows to custom controls.

To use the row drag and drop feature in Grid control, you need to inject the **RowDD** module in the grid. The **RowDD** is responsible for handling the row drag and drop functionality in the grid control. Once you have injected the **RowDD**, you can then use the [allowRowDragAndDrop](#) and [targetID](#) properties to enable and configure the row drag and drop feature in the Grid.

Drag and drop within grid

The drag and drop feature allows you to rearrange rows within the grid by dragging them using a drag icon. This feature can be enabled by setting the [allowRowDragAndDrop](#) property to **true**. This property is a boolean value that determines whether row drag and drop is enabled or not. By default, it is set to **false**, which means that row drag and drop is disabled.

Here's an example of how to enable drag and drop within the Grid:

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.RowDD);
var grid = new ej.grids.Grid({

```

```

dataSource: data,
allowRowDragAndDrop: true,
selectionSettings: { type: 'Multiple' },
height: 400,
columns: [
    { field: 'OrderID', headerText: 'Order ID', width: 120, textAlign:
'Right' },
    { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
    { field: 'OrderDate', headerText: 'Order Date', width: 100,
format: 'yMd', textAlign: 'Right' },
    { field: 'Freight', headerText: 'Freight', width: 120, format: 'C2',
textAlign: 'Right' },
    { field: 'ShipCity', headerText: 'Ship City', width: 130 },
    { field: 'ShipCountry', headerText: 'Ship Country', width: 130 }
]
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en">
<head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <div id="container">

```

```

        <div style="display: inline-block">
            <div id="Grid"></div>
        </div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Drag and drop to grid

The grid row drag and drop allows you to drag grid rows and drop to another grid. This feature can be enabled by setting the [allowRowDragAndDrop](#) property to **true** in the Grid control. This property specifies whether to enable or disable the row drag and drop feature in the Grid. By default, this property is set to **false**, which means that row drag and drop functionality is not enabled.

To specify the target control where the grid rows should be dropped, use the [targetID](#) property of the [rowDropSettings](#) object. The [targetID](#) property takes the ID of the target control as its value.

Here's an example code snippet that demonstrates how to enable Row drag and drop another Grid control:

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.RowDD);
var grid = new ej.grids.Grid({
    dataSource: data,
    allowRowDragAndDrop: true,
    rowDropSettings: { targetID: 'DestGrid' },
    selectionSettings: { type: 'Multiple' },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'ShipCity', headerText: 'Ship City', width: 150 },
        { field: 'ShipName', headerText: 'Ship Name', width: 150 }
    ]
});
grid.appendTo('#Grid');
var destGrid = new ej.grids.Grid({
    dataSource: [],
    allowRowDragAndDrop: true,
    rowDropSettings: { targetID: 'Grid' },
    selectionSettings: { type: 'Multiple' },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'ShipCity', headerText: 'Ship City', width: 150 },
        { field: 'ShipName', headerText: 'Ship Name', width: 150 }
    ]
});
destGrid.appendTo('#DestGrid');

```


INDEX.HTML

```

<!DOCTYPE html><html lang="en">
<head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container">
    <div style="display: inline-block">
      <div id="Grid"></div>
      <div id="DestGrid"></div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

* The row drag and drop feature is not supported in virtual scrolling and frozen rows and columns mode.

* In order to use row drag and drop, you need to inject the **RowDD** module in the grid.

Drag and drop to custom control

The Grid provides the feature to drag and drop grid rows to any custom control. This feature allows you to easily move rows from one control to another without having to manually copy and paste data. To enable row drag and drop, you need to set the [allowRowDragAndDrop](#) property to **true** and defining the custom control ID in the [targetID](#) property of the `rowDropSettings` object. The ID provided in `targetID` should correspond to the ID of the target control where the rows are to be dropped.

In the below example, the selected grid row is dragged and dropped in to the TreeGrid control by using [rowDrop](#) event. Once the row is dropped into the TreeGrid control, we have removed the corresponding grid row from grid and its data inserted in custom control.

INDEX.JS

```
ej.grids.Grid.Inject(ej.grids.RowDD);
var grid = new ej.grids.Grid({
  dataSource: sampleGridData,
  allowPaging: true,
  pageSettings: true,
  allowSelection: true,
  allowRowDragAndDrop: true,
  rowDropSettings: { targetID: 'TreeGrid' },
  selectionSettings: { type: 'Multiple' },
  editSettings: { allowDeleting: true },
  rowDrop: function (args) {
    var grid = document.getElementById('Grid').ej2_instances[0];
    var tree = document.getElementById('TreeGrid').ej2_instances[0];
    if (args.target.closest('.e-treegrid')) {
      args.cancel = true;
      var rowIndex =
        args.target.closest('.e-row') !== null
          ? args.target.closest('.e-row').rowIndex
          : 0;
      for (var i = 0; i < args.data.length; i++) {
        tree.addRecord(args.data[i], rowIndex);
        grid.deleteRecord('taskID', args.data[i]);
      }
    }
  },
  columns: [
    { field: 'taskID', headerText: 'taskID', textAlign: 'Right', width: 90 },
    { field: 'taskName', headerText: 'taskName', textAlign: 'Left', width: 180 },
    { field: 'description', headerText: 'description', textAlign: 'Left', width: 180 },
    { field: 'category', headerText: 'category', textAlign: 'Left', width: 180 },
    { field: 'startDate', headerText: 'startDate', textAlign: 'Right', format: 'yMd', width: 120 },
    { field: 'duration', headerText: 'duration', textAlign: 'Right', width: 80 }
  ],
});
grid.appendTo('#Grid');
```

```

var treeGridObj = new ej.treegrid.TreeGrid({
  childMapping: 'subtasks',
  editSettings: { allowAdding: true, allowEditing: true },
  columns: [
    { field: 'taskID', headerText: 'Task ID', width: 90, textAlign: 'Right' },
    { field: 'taskName', headerText: 'Task Name', width: 180, textAlign: 'Left' },
    { field: 'startDate', headerText: 'Start Date', width: 90, textAlign: 'Right', type: 'date', format: 'yMd' },
    { field: 'duration', headerText: 'Duration', width: 80, textAlign: 'Right' }
  ],
});
treeGridObj.appendTo('#TreeGrid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en">
<head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-calendars/styles/material.css" rel="stylesheet">
  <link href="http://cdn.syncfusion.com/ej2/ej2-treegrid/styles/material.css" rel="stylesheet" type="text/css"/>
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js" type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type="text/javascript"></script>
</head>
<body>
  <div id="container">

```

```

        <div style="display: inline-block">
            <div id="Grid"></div>
            <div id="TreeGrid"></div>
        </div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

* The **rowDrop** event is fired when a row is dropped onto a custom control, regardless of whether the drop is successful or not. You can use the **args.cancel** property to prevent the default action.

Drag and drop rows without drag icon

By default, when performing a drag and drop operation in the Syncfusion Grid, drag icons are displayed. However, in some cases, you may want to hide these drag icons. This can be achieved by setting the **targetID** property of the **rowDropSettings** object to the current Grid's ID.

By setting the **targetID**, the Grid will render without a helper icon for row dragging. You can then customize the drag and drop action by binding to the **rowDrop** event of the Grid. In the **rowDrop** event, you can prevent the default action by setting **args.cancel** to **true**, and reorder the rows using the **reorderRows** method of the Grid.

Here's an example of how to hide the drag and drop icon in the Syncfusion Grid:

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.RowDD);
var grid = new ej.grids.Grid({
    dataSource: data,
    allowRowDragAndDrop: true,
    rowDropSettings: { targetID: 'Grid' },
    selectionSettings: { type: 'Multiple' },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', width: 120, textAlign:
'Right' },
        { field: 'CustomerID', headerText: 'Customer Name', width: 130 },
        { field: 'OrderDate', headerText: 'Order Date', width: 120, format:
'yMd', textAlign: 'Right' },
        { field: 'Freight', headerText: 'Freight', width: 120, format:
'C2', textAlign: 'Right' },
        { field: 'ShipCity', headerText: 'Ship City', width: 130 },
        { field: 'ShipCountry', headerText: 'Ship Country', width: 130 },
    ],
    rowDrop: (args) => {
        args.cancel = true;
        var value = [];
        for (var index = 0; index < args.rows.length; index++) {
            value.push(args.fromIndex + index);
        }
        grid.reorderRows(value, args.dropIndex);
    }
});

```

```
    },
  });
  grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container">
    <div style="display: inline-block">
      <div id="Grid"></div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

* The selection feature must be enabled in the Grid to allow users to select rows before performing the drag and drop operation.

* Multiple rows can be selected by clicking and dragging inside the grid. For multiple row selection, the [type](#) property must be set to **Multiple**.

Drag and drop events

The Grid control provides a set of events that are triggered during drag and drop operations on grid rows. These events allow you to customize the drag element, track the progress of the dragging operation, and perform actions when a row is dropped on a target element. The following events are available:

1. [rowDragStartHelper](#): This event is triggered when a click occurs on the drag icon or a grid row. It allows you to customize the drag element based on specific criteria.
2. [rowDragStart](#): This event is triggered when the dragging of a grid row starts.
3. [rowDrag](#): This event is triggered continuously while the grid row is being dragged.
4. [rowDrop](#): This event is triggered when a drag element is dropped onto a target element.

INDEX.JS

```
ej.grids.Grid.Inject(ej.grids.RowDD);
var grid = new ej.grids.Grid({
  dataSource: data,
  allowPaging: true,
  allowRowDragAndDrop: true,
  selectionSettings: { type: 'Multiple' },
  columns: [
    { field: 'OrderID', headerText: 'Order ID', width: 120, textAlign:
'Right' },
    { field: 'CustomerID', headerText: 'Customer ID', width: 130 },
    { field: 'OrderDate', headerText: 'Order Date', width: 120, format:
'yMd', textAlign: 'Right' },
    { field: 'Freight', headerText: 'Freight', width: 120, format: 'C2',
textAlign: 'Right' }
  ],
  rowDragStartHelper: function (args) {
    document.getElementById('message').innerText = `rowDragStartHelper
event triggered`;
    if (args.data[0].OrderID === 10248) {
      args.cancel = true;
    }
  },
  rowDragStart: function (args) {
    document.getElementById('message').innerText = `rowDragStart event
triggered`;
    args.cancel = true;
  },
  rowDrag: function (args) {
    document.getElementById('message').innerText = `rowDrag event
triggered`;
    args.rows.forEach((row) => {
      row.classList.add('drag-limit');
    });
  },
  rowDrop: function (args) {
```

```

        document.getElementById('message').innerText = `rowDrop event
triggered`;
        var value = [];
        for (var index = 0; index < args.rows.length; index++) {
            value.push(args.fromIndex + index);
        }
        grid.reorderRows(value, args.dropIndex);
    },
    });
    grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <style>
    .event-message {
      text-align: center;
      color: red;
      font-size: 16px;
    }
    .drag-limit .e-rowcell {
      border: 1px solid red;
    }
  </style>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>

```

```
<body>
  <div id="container">
    <p class="event-message" id="message"></p>
    <div id="Grid"></div>
  </div>
</body>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Limitations

- Single row is able to drag and drop in same grid without enable the row selection.
- Row drag and drop feature is not having built in support with row template, detail template, hierarchy grid and grouping features of grid.

See also

[Sorting data in the Syncfusion Grid](#)

[Filtering data in the Syncfusion Grid](#)

Row spanning in ##Platform_Name## Grid control

The grid provides an option to span row cells, allowing you to merge two or more cells in a row into a single cell. This feature can be useful in scenarios where you want to display information that spans across multiple rows, but want to avoid repeating the same information in each row.

To achieve this, You need to define the [rowSpan](#) attribute to span cells in the [queryCellInfo](#) event. The `rowSpan` attribute is used to specify the number of rows that the current cell should span.

The `queryCellInfo` event is triggered for each cell in the grid, and allows you to customize the cells in the grid. By handling this event, you can set the `rowSpan` attribute for a cell to achieve row spanning.

In the following demo, **Davolio** cell is spanned to two rows in the **EmployeeName** column. Also Grid supports the spanning of rows and columns for same cells. **Lunch Break** cell is spanned to two rows and three columns in the **1:00** column.

INDEX.JS

```
var grid = new ej.grids.Grid({
  dataSource: columnSpanData,
  queryCellInfo: QueryCellEvent,
  gridLines: 'Both',
  columns: [
    { field: 'EmployeeID', headerText: 'Employee ID', isPrimaryKey:
true, textAlign: 'Right', width: 150 },
    { field: 'EmployeeName', headerText: 'Employee Name', width: 200 },
    { field: '9:00', headerText: '9.00 AM', width: 120 },
    { field: '9:30', headerText: '9.30 AM', width: 120 },
    { field: '10:00', headerText: '10.00 AM', width: 120 },
    { field: '10:30', headerText: '10.30 AM', width: 120 },
    { field: '11:00', headerText: '11.00 AM', width: 120 },
```



```

        { field: '11:30', headerText: '11.30 AM', width: 120 },
        { field: '12:00', headerText: '12.00 PM', width: 120 },
        { field: '12:30', headerText: '12.30 PM', width: 120 },
        { field: '1:00', headerText: '1.00 PM', width: 120 },
        { field: '1:30', headerText: '1.30 PM', width: 120 },
        { field: '2:00', headerText: '2.00 PM', width: 120 },
        { field: '2:30', headerText: '2.30 PM', width: 120 },
        { field: '3:00', headerText: '3.00 PM', width: 120 },
        { field: '3:30', headerText: '3.30 PM', width: 120 },
        { field: '4:00', headerText: '4.00 PM', width: 120 },
        { field: '4:30', headerText: '4.30 PM', width: 120 },
        { field: '5:00', headerText: '5.00 PM', width: 120 }
    ],
    width: 'auto',
    height: 300,
    allowTextWrap: true
});
grid.appendTo('#Grid');
function QueryCellEvent(args) {
    var data = args.data;
    switch (data.EmployeeID) {
        case 10001:
            if (args.column.field === '9:00' || args.column.field === '2:30'
|| args.column.field === '4:30') {
                args.colSpan = 2;
            } else if (args.column.field === '11:00') {
                args.colSpan = 3;
            } else if (args.column.field === 'EmployeeName') {
                args.rowSpan = 2;
            } else if (args.column.field === '1:00') {
                args.colSpan = 3;
                args.rowSpan = 2;
            }
            break;
        case 10002:
            if (args.column.field === '9:30' || args.column.field === '2:30'
||
                args.column.field === '4:30') {
                args.colSpan = 3;
            } else if (args.column.field === '11:00') {
                args.colSpan = 4;
            }
            break;
        case 10003:
            if (args.column.field === '9:00' || args.column.field ===
'11:30') {
                args.colSpan = 3;
            } else if (args.column.field === '10:30' || args.column.field
=== '3:30' ||
                args.column.field === '4:30' || args.column.field ===
'2:30') {
                args.colSpan = 2;
            }
            break;
        case 10004:
            if (args.column.field === '9:00') {
                args.colSpan = 3;
            }
    }
}

```

```

        } else if (args.column.field === '11:00') {
            args.colSpan = 4;
        } else if (args.column.field === '4:00' || args.column.field ===
'2:30') {
            args.colSpan = 2;
        }
        break;
    case 10005:
        if (args.column.field === '9:00') {
            args.colSpan = 4;
        } else if (args.column.field === '11:30') {
            args.colSpan = 3;
        } else if (args.column.field === '3:30' || args.column.field ===
'4:30' || args.column.field === '2:30') {
            args.colSpan = 2;
        }
        break;
    case 10006:
        if (args.column.field === '9:00' || args.column.field === '4:30'
||
            args.column.field === '2:30' || args.column.field ===
'3:30') {
            args.colSpan = 2;
        } else if (args.column.field === '10:00' || args.column.field
=== '11:30') {
            args.colSpan = 3;
        }
        break;
    case 10007:
        if (args.column.field === '9:00' || args.column.field === '3:00'
|| args.column.field === '10:30') {
            args.colSpan = 2;
        } else if (args.column.field === '11:30' || args.column.field
=== '4:00') {
            args.colSpan = 3;
        }
        break;
    case 10008:
        if (args.column.field === '9:00' || args.column.field ===
'10:30' || args.column.field === '2:30') {
            args.colSpan = 3;
        } else if (args.column.field === '4:00') {
            args.colSpan = 2;
        }
        break;
    case 10009:
        if (args.column.field === '9:00' || args.column.field ===
'11:30') {
            args.colSpan = 3;
        } else if (args.column.field === '4:30' || args.column.field ===
'2:30') {
            args.colSpan = 2;
        }
        break;
    case 10010:
        if (args.column.field === '9:00' || args.column.field === '2:30'
||

```

```

        args.column.field === '4:00' || args.column.field ===
'11:30') {
        args.colSpan = 3;
    } else if (args.column.field === '10:30') {
        args.colSpan = 2;
    }
    break;
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en">
<head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>

```

```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

To disable the spanning for particular grid page, you need to use **requestType** from **queryCellInfo** event argument.

The **rowSpan** and **colSpan** attributes can be used together to merge cells both vertically and horizontally.

Cell

Cell in **##Platform_Name##** Grid control

Cell customization

The appearance of cells can be customized by using the [queryCellInfo](#) event. The [queryCellInfo](#) event triggers for every cell. In that event handler, you can get [QueryCellInfoEventArgs](#) that contains the details of the cell.

INDEX.JS

```
var grid = new ej.grids.Grid({
  dataSource: data,
  enableHover: false,
  allowSelection: false,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
    { field: 'OrderDate', headerText: 'Order Date', width: 100, format:
'yMd'},
    { field: 'Freight', headerText: 'Freight', width: 100, format:
'C2'},
    { field: 'ShipCity', headerText: 'Ship City', width: 100 },
  ],
  height: 315,
  queryCellInfo: customiseCell
});
grid.appendTo('#Grid');
function customiseCell(args) {
  if(args.column.field === 'Freight') {
    if (args.data['Freight'] < 30){
      args.cell.classList.add('below-30');
    } else if (args.data['Freight'] < 80 ) {
      args.cell.classList.add('below-80');
    } else {
      args.cell.classList.add('above-80');
    }
  }
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```

<meta name="description" content="Typescript Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Custom attributes

You can customize the grid cells by adding a CSS class to the [customAttribute](#) property of the column.

```

`css
.e-attr {

```

```
background: '#d7f0f4';
}
,
`ts
{
field: "ShipCity", headerText: "Ship City", customAttributes: {class: "e-attr"}, width: "120"
}
,

```

In the below example, we have customized the cells of **OrderID** and **ShipCity** columns.

INDEX.JS

```
var grid = new ej.grids.Grid({
  dataSource: data,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', customAttributes:
{class: "e-attr"}, textAlign: 'Right', width: 100 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
    { field: 'ShipCity', headerText: 'Ship City', customAttributes:
{class: "e-attr"}, width: 100 },
    { field: 'ShipName', headerText: 'Ship Name', width: 100 }
  ],
  height: 280
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
.e-attr {
    background: #d7f0f4;
}
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Grid lines

The [gridLines](#) have option to display cell border and it can be defined by the [gridLines](#) property.

The available modes of grid lines are:

- | Modes | Actions |
|-------------|---|
| ----- ----- | |
| Both | Displays both the horizontal and vertical grid lines. |
| None | No grid lines are displayed. |
| Horizontal | Displays the horizontal grid lines only. |
| Vertical | Displays the vertical grid lines only. |
| Default | Displays grid lines based on the theme. |

INDEX.JS

```

var grid = new ej.grids.Grid({
  dataSource: data,
  gridLines: 'Both',
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
    { field: 'ShipCity', headerText: 'Ship City', width: 100 },
    { field: 'ShipName', headerText: 'Ship Name', width: 100 }
  ],
  height: 315
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>

```



```

<body>
  <script id="rowtemplate" type="text/x-template">
    <tr>
      <td class="photo">
        
      </td>
      <td class="details">
        <table class="CardTable" cellpadding="3" cellspacing="2">
          <colgroup>
            <col width="50%">
            <col width="50%">
          </colgroup>
          <tbody>
            <tr>
              <td class="CardHeader">First Name </td>
              <td>{FirstName} </td>
            </tr>
            <tr>
              <td class="CardHeader">Last Name</td>
              <td>{LastName} </td>
            </tr>
            <tr>
              <td class="CardHeader">Title
              </td>
              <td>{Title}
              </td>
            </tr>
            <tr>
              <td class="CardHeader">Country
              </td>
              <td>{Country}
              </td>
            </tr>
          </tbody>
        </table>
      </td>
    </tr>
  </script>

  <div id="container">
    <div id="Grid"></div>
  </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

By default, the grid renders with **Default** mode.

[See Also](#)

Content in `##Platform_Name##` Grid control

The HTML tags can be displayed in the Grid header and content by enabling the [disableHtmlEncode](#) property.

INDEX.JS

```
var grid = new ej.grids.Grid({
  dataSource: data,
  columns: [
    { field: 'OrderID', headerText: '<span> Order ID </span>',
      disableHtmlEncode: true, textAlign: 'Right', width: 140 },
    { field: 'CustomerID', headerText: '<span> Customer ID </span>',
      disableHtmlEncode: false, width: 120 },
    { field: 'OrderDate', headerText: 'Order Date', width: 100, format:
      'yMd' },
    { field: 'ShipCity', headerText: 'Ship City', width: 100 },
  ],
  height: 315
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Auto wrap in ##Platform_Name## Grid control

The auto wrap allows the cell content of the grid to wrap to the next line when it exceeds the boundary of the cell width. The Cell Content wrapping works based on the position of white space between words. To enable auto wrap, set the [allowTextWrap](#) property to **true**. You can configure the auto wrap mode by setting the [textWrapSettings.wrapMode](#) property.

There are three types of [wrapMode](#). They are:

- **Both:** Both value is set by default. Auto wrap will be enabled for both the content and the header.
- **Header:** Auto wrap will be enabled only for the header.
- **Content:** Auto wrap will be enabled only for the content.

Note: When a column width is not specified, then auto wrap of columns will be adjusted with respect to the grid's width.

In the following example, the [textWrapSettings.wrapMode](#) is set to **Content**.

INDEX.JS

```

var grid = new ej.grids.Grid({
    dataSource: data,
    gridLines: 'Default',
    allowTextWrap: true,
    textWrapSettings: { wrapMode: 'Content' },
    columns: [
        { field: 'RoolNo', headerText: 'Rool No', width: 120 },
        { field: 'Name', headerText: 'Name of the inventor', width: 100 },
        { field: 'patentfamilies', headerText: 'No of patentfamilies',
width: 100 },
        { field: 'Country', headerText: 'country', width: 130 },

```

```

        { field: 'mainfields', headerText: 'Main fields of Invention',
width: 150 },
        ],
        height: 315
    });
    grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Grid"></div>
  </div>
<script>
var ele = document.getElementById('container');

```

```

if(ele) {
    ele.style.visibility = "visible";
}

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Clip mode in ##Platform_Name## Grid control

The clip mode provides options to display its overflow cell content and it can be defined by the [columns.clipMode](#) property.

There are three types of [clipMode](#). They are:

- **Clip:** Truncates the cell content when it overflows its area.
- **Ellipsis:** Displays ellipsis when the cell content overflows its area.
- **EllipsisWithTooltip:** Displays ellipsis when the cell content overflows its area, also it will display the tooltip while hover on ellipsis is applied.

INDEX.JS

```

var grid = new ej.grids.Grid({
    dataSource: data,
    allowPaging: true,
    columns: [
        { field: 'RoolNo', headerText: 'Rool No', width: 100 },
        { field: 'Name', headerText: 'Name of the inventor', clipMode:
'Clip', width: 100 },
        { field: 'patentfamilies', headerText: 'No of patent families',
clipMode: 'Ellipsis', width: 100 },
        { field: 'Country', headerText: 'Country', width: 80 },
        { field: 'mainfields', headerText: 'Main fields of Invention',
clipMode: 'EllipsisWithTooltip', width: 100 },
    ]
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

By default, [columns.clipMode](#) value is **Ellipsis**.

Editing

Edit in ##Platform_Name## Grid control

The Grid component has options to dynamically insert, delete and update records. Editing feature requires a primary key column for CRUD operations. To define the primary key, set [columns.isPrimaryKey](#) to **true** in particular column.

You can start the edit action either by double clicking the particular row or by selecting the required row and click on **Edit** button in the toolbar. Similarly, you can add a new record to grid either by clicking on **Add** button in the toolbar or on an external button which is bound to invoke the [addRecord](#) method of the grid, **Save** and **Cancel** while in edit mode is possible using respective toolbar icon in grid.

Deletion of the record is possible by selecting the required row and click on **Delete** button in the toolbar.

To use CRUD, inject the [Edit](#) module in grid.

INDEX.JS

```
ej.grids.Grid.Inject(ej.grids.Edit);
var grid = new ej.grids.Grid({
  dataSource: data,
  editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true },
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100, isPrimaryKey: true },
    { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
    { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C2' },
    { field: 'ShipCountry', headerText: 'Ship Country', width: 150 }
  ],
  height: 315
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <style>
```

```

        .e-row[aria-selected="true"] .e-customizedExpandcell {
            background-color: #e0e0e0;
        }
        .e-grid.e-gridhover tr[role='row']:hover {
            background-color: #eee;
        }
        .e-expand::before {
            content: '\e5b8';
        }
        .empImage {
            margin: 6px 16px;
            float: left;
            width: 50px;
            height: 50px;
        }
    </style>
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
    </head>
    <body>

        <div id="container">
            <div id="Grid"></div>
        </div>
    <script>
    var ele = document.getElementById('container');
    if(ele) {
        ele.style.visibility = "visible";
    }
    </script>
    <script src="index.js" type="text/javascript"></script>
    </body></html>

```

* If [columns.isIdentity](#) is enabled, then it will be considered as a read-only column when editing and adding a record.

* You can disable editing for a particular column, by specifying [columns.allowEditing](#) to `false`.

Toolbar with edit option

The grid toolbar has the [built-in items](#) to execute Editing actions. You can define this by using the [toolbar](#) property.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Edit, ej.grids.Toolbar);
var grid = new ej.grids.Grid({
    dataSource: data,
    toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
    editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
    true },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
        width: 100, isPrimaryKey: true },

```



```

        { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C2' },
        { field: 'ShipCountry', headerText: 'Ship Country', width: 150 }
    ],
    height: 273
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
      background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
      background-color: #eee;
    }
    .e-expand::before {
      content: '\e5b8';
    }
    .empImage {
      margin: 6px 16px;
    }
  </style>

```

```

        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Disable editing for particular column

You can disable editing for particular columns by using the [columns.allowEditing](#).

In the following demo, editing is disabled for the **CustomerID** column.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Edit, ej.grids.Toolbar);
var grid = new ej.grids.Grid({
    dataSource: data,
    toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
    editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100, isPrimaryKey: true },
        { field: 'CustomerID', headerText: 'Customer ID', width: 120,
allowEditing: false },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
editType: 'numericedit', width: 120, format: 'C2' },
        { field: 'ShipCountry', headerText: 'Ship Country', editType:
'dropdownedit', width: 150 }
    ],
    height: 273
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
<title>EJ2 Grid</title>

```

```

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
.e-row[aria-selected="true"] .e-customizedExpandcell {
    background-color: #e0e0e0;
}
.e-grid.e-gridhover tr[role='row']:hover {
    background-color: #eee;
}
.e-expand::before {
    content: '\e5b8';
}
.empImage {
    margin: 6px 16px;
    float: left;
    width: 50px;
    height: 50px;
}
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

<div id="container">

```

```

        <div id="Grid"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Disable editing for a particular row or cell

You can disable the editing for a particular row by using the [actionBegin](#) event of Grid based on **requestType** as **beginEdit**.

In the below demo, the rows which are having the value for **ShipCountry** column as "France" is prevented from editing.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Edit, ej.grids.Toolbar);
var grid = new ej.grids.Grid({
    dataSource: data,
    actionBegin: function(args) {
        if (args.requestType === "beginEdit") {
            if (args.rowData.ShipCountry == "France") {
                args.cancel = true;
            }
        }
    },
    toolbar: ['Edit', 'Update', 'Cancel'],
    editSettings: { allowEditing: true },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100, isPrimaryKey: true },
        { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
editType: 'numericedit', width: 120, format: 'C2' },
        { field: 'ShipCountry', headerText: 'Ship Country', editType:
'dropdownedit', width: 150 }
    ],
    height: 273
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {

```

```

    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

For batch mode of editing, you can use [cellEdit](#) event of Grid. In the below demo, the cells which are having the value as "France" is prevented from editing.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Edit, ej.grids.Toolbar);
var grid = new ej.grids.Grid({
  dataSource: data,
  cellEdit: function(args) {
    if (args.value == "France") {
      args.cancel = true;
    }
  },
  toolbar: ['Edit', 'Update', 'Cancel'],
  editSettings: { allowEditing: true, mode: 'Batch' },
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100, isPrimaryKey: true },
    { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
    { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
editType: 'numericedit', width: 120, format: 'C2' },
    { field: 'ShipCountry', headerText: 'Ship Country', editType:
'dropdownedit', width: 150 }
  ],
  height: 273
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Editing template column

You can edit the template column value by defining the **field** for that particular column.

In the below demo, the **ShipCountry** column is rendered with the template.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Edit, ej.grids.Toolbar);
var grid = new ej.grids.Grid({
  dataSource: data,
  toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
  editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true },
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100, isPrimaryKey: true },
    { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
    { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
editType: 'numericedit', width: 120, format: 'C2' },
    { field: 'ShipCountry', headerText: 'Ship Country', template:
'#template', editType: 'dropdownedit', width: 150 }
  ],
  height: 273
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

```



```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <script id="template" type="text/x-template">
            <a href="#">${ShipCountry}</a>
        </script>
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Troubleshoot editing works only for first row

The Editing functionalities can be performed based upon the primary key value of the selected row. If **primaryKey** is not defined in the grid, then edit or delete action take places the first row.

How to make a Grid column always editable

Make the Grid column always editable using the column template feature of the Grid.

In the following example, the textbox is rendered in the Freight column using a column template. The keyup event for the Grid is bound using the [created](#) event of the Grid, and the edited changes are saved in the data source using the [updateRow](#) method of the Grid.

INDEX.JS

```

var grid = new ej.grids.Grid({
    dataSource: data,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, isPrimaryKey: true },
        { field: 'OrderDate', headerText: 'Order Date', width: 130,
textAlign: 'Right', format: 'yMd' },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C2' },
        { field: 'ShipCountry', headerText: 'Ship Country', width: 140 },
        { field: 'Freight', headerText: 'Receipt Amount', width: 150,
template: '#template' }
    ],
    height: 315,
    created: (args) => {
        grid.element.addEventListener('keyup', function (e) { // Bind the
keyup event for the grid.
            if (e.target.classList.contains('custemp')) { // Based on this
condition, you can find whether the target is an input element or not.

```

```

        var row = ej.grids.parentsUntil(e.target, 'e-row');
        var rowIndex = row.rowIndex; // Get the row index.
        var uid = row.getAttribute('data-uid');
        var rowData = grid.getRowObjectFromUID(uid).data; // Get the
row data.
        rowData.Freight = e.target.value; // Update the new value
for the corresponding column.
        grid.updateRow(rowIndex, rowData); // Update the modified
value in the row data.
    }
    });
}
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```

```

    <script id="template" type="text/x-template">
        <input id='${OrderID}' value='${Freight}' class='custemp'
type='text' style='width: 100%'>
    </script>
    <div id="container">
        <div id="Grid"></div>
    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Customize dialog when editing](#)
- [Cascading DropDownList with Grid Editing](#)
- [Customize the Edit Dialog](#)
- [Tab Inside the Dialog Template](#)
- [How to use Document Editor as an edit field in Data Grid](#)
- [How to render custom confirmation dialog while updating edit operations](#)
- [How to render ColorPicker component for particular column while editing a record](#)
- [How to positioning the validation error message](#)

Edit types in `##Platform_Name##` Grid control

Customize editors using params

The [columns.editType](#) is used to define the editor component for any particular column. You can set the [columns.editType](#) based on data type of the column.

- [NumericTextBox](#) component for integers, double, and decimal data types.
- [TextBox](#) component for string data type.
- [DropDownList](#) component to show all unique values related to that field.
- [CheckBox](#) component for boolean data type.
- [DatePicker](#) component for date data type.
- [DateTimePicker](#) component for date time data type.

Also, you can customize the behavior of the editor component through the [columns.edit.params](#).

The following table describes editor component and their example edit params of the column.

Component | Example

[NumericTextBox](#) | params: { decimals: 2, value: 5 }

[DropDownList](#) | params: { value: 'Germany' }

[Checkbox](#) | params: { checked: true }

[DatePicker](#) | params: { format: 'dd.MM.yyyy' }

[DateTimePicker](#) | params: { value: new Date() }

INDEX.JS

```
ej.grids.Grid.Inject(ej.grids.Edit, ej.grids.Toolbar);
var grid = new ej.grids.Grid({
  dataSource: data,
  toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
  editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true },
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100, isPrimaryKey: true },
    { field: 'CustomerName', headerText: 'CustomerName', textAlign:
'Right', width: 200 },
    { field: 'OrderDate', headerText: 'Order Date', type: 'date', width:
120, format: { type: 'date', format: 'dd.MM.yyyy' }, editType:
'datepickeredit', edit: { params: { format: 'dd.MM.yy' } } },
    { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
editType: 'numericedit', width: 120, format: 'C2', edit: { params: {
decimals: 2, value: 5 } } },
    { field: 'ShipCountry', headerText: 'Ship Country', editType:
'dropdownedit', width: 150, edit: { params: { value: 'Germany' } } },
    { field: 'Verified', displayAsCheckBox: true, editType:
'booleanedit', textAlign: 'Center', width: 100, edit: { params: { checked:
true } } }
  ],
  height: 273
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

If edit type is not defined in the column, then it will be considered as the **stringedit** type (Textbox component) .

[Restrict to type decimal points in a NumericTextBox while editing the numeric column](#)

By default, the number of decimal places will be restricted to two in the NumericTextBox while editing the numeric column. We can restrict to type the decimal points in a NumericTextBox by using the **validateDecimalOnType** and **decimals** properties of NumericTextBox.

In the below demo, while editing the row we have restricted to type the decimal point value in the NumericTextBox of **Freight** column.

INDEX.JS

```
ej.grids.Grid.Inject(ej.grids.Edit, ej.grids.Toolbar);
var grid = new ej.grids.Grid({
  dataSource: data,
  toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
  editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true },
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 150, isPrimaryKey: true },
    { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
    {
      field: 'Freight', headerText: 'Freight', textAlign: 'Right',
editType: 'numericedit', width: 150, edit: {
        params: {
          validateDecimalOnType: true,
          decimals: 0,
          format: "N"
        }
      }
    },
    { field: 'ShipCity', headerText: 'Ship City', editType:
'dropdownedit', width: 150 }
  ],
  height: 265
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <script id="template" type="text/x-template">
            <a href="#">${ShipCountry}</a>
        </script>
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Provide custom data source and enabling filtering to DropDownList

You can provide data source to the DropDownList by using the [columns.edit.params](#) property.

While setting new data source using edit params, you must specify a new `query` property too for the dropdownlist as follows,

```

`ts
{
  params: {
    query: new Query(),
    dataSource: country,
    fields: { value: 'ShipCountry', text: 'ShipCountry' },
    allowFiltering: true
  }
}

```

```
}
,
```

You can also enable filtering for the dropdownlist by passing the `allowFiltering` as `true` to the edit params.

In the below demo, DropDownList is rendered with custom Datasource for the `ShipCountry` column and enabled filtering to search dropdownlist items.

INDEX.JS

```
var country = [
    { ShipCountry: 'United States', countryId: '1' },
    { ShipCountry: 'Australia', countryId: '2' },
    { ShipCountry: 'India', countryId: '2' }
];
ej.grids.Grid.Inject(ej.grids.Edit, ej.grids.Toolbar);
var grid = new ej.grids.Grid({
    dataSource: cascadeData,
    toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
    editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true, mode: 'Normal' },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100, isPrimaryKey: true, validationRules: { required: true } },
        { field: 'CustomerID', headerText: 'Customer ID', width: 120,
validationRules: { required: true, minLength: 3 } },
        { field: 'ShipCountry', headerText: 'Ship Country', width: 120,
editType: 'dropdownedit', edit: {
            params: {
                query: new ej.data.Query(),
                dataSource: country,
                fields: { value: 'ShipCountry', text: 'ShipCountry' },
                allowFiltering: true
            }
        }
    ]
},
height: 273
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>

```

```
</body></html>
```

Open popup while focusing the edit dropdown list

You can open the DropDownList popup while focusing the cell by invoking the [showPopup](#) method inside the ['focus'](#) event of DropDownList component.

In the below demo, we have bound the focus event for the edit DropDownList using the [columns.edit.params](#) property.

INDEX.JS

```
ej.grids.Grid.Inject(ej.grids.Edit, ej.grids.Toolbar);
var grid = new ej.grids.Grid({
  dataSource: data,
  toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
  editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true },
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100, isPrimaryKey: true },
    { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
    { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
editType: 'numericedit', width: 120, format: 'C2' },
    { field: 'ShipCountry', headerText: 'Ship Country', edit: { params:
{ focus: ddFocus } }, editType: 'dropdownedit', width: 150 }
  ],
  height: 265
});
grid.appendTo('#Grid');
function ddFocus(e) {
  e.event.target.querySelector('.e-
dropdownlist').ej2_instances[0].showPopup();
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Custom editors using template

The cell edit template is used to add a custom component for a particular column by invoking the following functions:

- **create** - It is used to create the element at the time of initialization.
- **write** - It is used to create the custom component or assign default value at the time of editing.
- **read** - It is used to read the value from the component at the time of save.
- **destroy** - It is used to destroy the component.

Render TimePicker component while editing

Use the cell edit template feature of the Grid to render the TimePicker component in the Grid edit form. In the below sample, we have rendered TimePicker component in the **OrderDate** column.

INDEX.JS

```
var ddElem;
var timeObject;
ej.base.enableRipple(true);
ej.grids.Grid.Inject(ej.grids.Edit, ej.grids.Toolbar, ej.grids.Page);
var grid = new ej.grids.Grid({
  dataSource: purchaseData,
  allowPaging: true,
  toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
  editSettings: { allowEditing: true, allowAdding: true, allowDeleting: true
},
  columns: [
    { field: 'OrderID', headerText: 'Order ID', type: 'number',
isPrimaryKey: true, validationRules: { required: true }, textAlign: 'Right',
width: 100 },
    { field: 'CustomerID', headerText: 'Customer ID', type: 'string', width:
140 },
    { field: 'Freight', headerText: 'Freight', type: 'number', editType:
'numericedit', format: 'C2', textAlign: 'Right', width: 120 },
    { field: 'OrderDate', headerText: 'Order Date', type: 'date', format:
'hh:mm', width: 150, edit: {
      create: createOrderDateFn,
      destroy: destroyOrderDateFn,
      read: readOrderDateFn,
      write: writeOrderDateFn }
    }
  ],
  pageSettings: { pageSize: 7 },
  height: 255,
});
grid.appendTo('#Grid');
function createOrderDateFn() {
  ddElem = document.createElement('input');
  return ddElem;
}
function destroyOrderDateFn() {
  timeObject.destroy();
}
function readOrderDateFn() {
  return timeObject.value;
}
function writeOrderDateFn(args) {
  timeObject = new ej.calendars.TimePicker({
    value: args.rowData[args.column.field],
    step: 60
```

```
});
timeObject.appendTo(ddElem);
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
      background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
      background-color: #eee;
    }
    .e-expand::before {
      content: '\e5b8';
    }
    .empImage {
      margin: 6px 16px;
      float: left;
      width: 50px;
      height: 50px;
    }
  </style>
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Render AutoComplete component while editing

Use the cell edit template feature of the Grid to render the AutoComplete component in the Grid edit form. In the below sample, we have rendered AutoComplete component in the **CustomerID** column.

INDEX.JS

```

var inpuEle;
var autoCompleteIns;
var autoCompleteData = [
    { CustomerID: 'VINET', Id: '1' },
    { CustomerID: 'TOMSP', Id: '2' },
    { CustomerID: 'HANAR', Id: '3' },
    { CustomerID: 'VICTE', Id: '4' },
    { CustomerID: 'SUPRD', Id: '5' }
];
ej.grids.Grid.Inject(ej.grids.Edit, ej.grids.Toolbar, ej.grids.Page);
var grid = new ej.grids.Grid({
    dataSource: purchaseData,
    allowPaging: true,
    toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
    editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', type: 'number',
isPrimaryKey: true, validationRules: { required: true }, textAlign: 'Right',
width: 100 },
        { field: 'CustomerID', headerText: 'Customer ID', type: 'string',
width: 140, edit: {
            create: createCustomerIDFn,
            destroy: destroyCustomerIDFn,
            read: readCustomerIDFn,
            write: writeCustomerIDFn }
        },
        { field: 'Freight', headerText: 'Freight', type: 'number', editType:
'numericedit', format: 'C2', textAlign: 'Right', width: 120 },

```

```

        { field: 'OrderDate', headerText: 'Order Date', type: 'date', format:
'yMd', editType: 'datepickeredit', width: 150 }
    ],
    pageSettings: { pageSize: 7 },
    height: 255,
});
grid.appendTo('#Grid');
function createCustomerIDFn() {
    inpuEle = document.createElement('input');
    return inpuEle;
}
function destroyCustomerIDFn() {
    autoCompleteIns.destroy();
}
function readCustomerIDFn() {
    return autoCompleteIns.value;
}
function writeCustomerIDFn(args) {
    autoCompleteIns = new ej.dropdowns.AutoComplete({
        allowCustom: true,
        value: args.rowData[args.column.field],
        dataSource: autoCompleteData,
        fields: { value: 'CustomerID', text: 'CustomerID' }
    });
    autoCompleteIns.appendTo(inpuEle);
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Render MultiSelect DropDown component while editing

Use the cell edit template feature of the Grid to render the MultiSelect DropDown component in the Grid edit form. In the below sample, we have rendered MultiSelect DropDown component in the **ShipCity** column.

INDEX.JS

```

var ddElem;
var multiSelectObj;
var multiselectDatasource = [
    { ShipCity: 'Reims', Id: '1' },
    { ShipCity: 'Münster', Id: '2' },
    { ShipCity: 'Rio de Janeiro', Id: '3' },

```



```

    { ShipCity: 'Lyon', Id: '4' },
    { ShipCity: 'Charleroi', Id: '5' },
  ];
  ej.grids.Grid.Inject(ej.grids.Edit, ej.grids.Toolbar, ej.grids.Page);
  var grid = new ej.grids.Grid({
    dataSource: purchaseData,
    allowPaging: true,
    toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
    editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true },
    columns: [
      { field: 'OrderID', headerText: 'Order ID', type: 'number',
isPrimaryKey: true, validationRules: { required: true }, textAlign: 'Right',
width: 100 },
      { field: 'CustomerID', headerText: 'Customer ID', type: 'string',
width: 140 },
      { field: 'Freight', headerText: 'Freight', type: 'number', editType:
'numericedit', format: 'C2', textAlign: 'Right', width: 120 },
      { field: 'ShipCity', headerText: 'Ship City', type: 'string', width:
180, edit: {
        create: createShipCityFn,
        read: readShipCityFn,
        destroy: destroyShipCityFn,
        write: writeShipCityFn }
      }
    ],
    pageSettings: { pageSize: 7 },
    height: 255,
  });
  grid.appendTo('#Grid');
  function createShipCityFn() {
    ddElem = document.createElement('input');
    return ddElem;
  }
  function readShipCityFn() {
    return multiSelectObj.value.join(', ');
  }
  function destroyShipCityFn() {
    multiSelectObj.destroy();
  }
  function writeShipCityFn(args) {
    {
      let multiSelectVal = args.rowData[args.column.field]
        ? args.rowData[args.column.field].split(',')
        : [];
      multiSelectObj = new ej.dropdowns.MultiSelect({
        value: multiSelectVal,
        dataSource: multiSelectDatasource,
        fields: { value: 'ShipCity', text: 'ShipCity' },
        floatLabelType: 'Never',
        mode: 'Box',
      });
      multiSelectObj.appendTo(ddElem);
    }
  }
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
      background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
      background-color: #eee;
    }
    .e-expand::before {
      content: '\e5b8';
    }
    .empImage {
      margin: 6px 16px;
      float: left;
      width: 50px;
      height: 50px;
    }
  </style>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>

```

```

</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
</body>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Render MaskedTextBox component while editing

Use the cell edit template feature of the Grid to render the MaskedTextBox component in the Grid edit form. In the following sample, the MaskedTextBox component is rendered in the Mask column.

INDEX.JS

```

var elem;
var maskObj;
ej.grids.Grid.Inject(ej.grids.Edit, ej.grids.Toolbar);
var grid = new ej.grids.Grid({
    dataSource: data,
    toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
    editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100, isPrimaryKey: true },
        { field: 'CustomerID', headerText: 'Customer ID', width: 120, },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
editType: 'numericedit', width: 120, format: 'C2' },
        {
            field: 'Mask', headerText: 'Mask', width: 140, edit: {
                create: function() {
                    elem = document.createElement('input');
                    return elem;
                },
                read: function() {
                    return maskObj.value;
                },
                destroy: function() {
                    maskObj.destroy();
                },
                write: function(args) {
                    maskObj = new ej.inputs.MaskedTextBox({
                        mask: "0-0-0-0",
                        value: args.rowData.Mask
                    });
                    maskObj.appendTo(args.element);
                }
            }
        }
    ]
});

```

```

    ],
    height: 273
  });
  grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
      background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
      background-color: #eee;
    }
    .e-expand::before {
      content: '\e5b8';
    }
    .empImage {
      margin: 6px 16px;
      float: left;
      width: 50px;
      height: 50px;
    }
  </style>

```

```

</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Render RichTextEditor component while editing

Use the cell edit template feature of the Grid to render the RichTextEditor component in the Grid edit form. In the below sample, we have rendered RichTextEditor component in the **ShipAddress** column, so we use [allowTextWrap](#) property to true.

INDEX.JS

```

var tbElem;
var textEditor;
ej.grids.Grid.Inject(ej.grids.Edit, ej.grids.Toolbar, ej.grids.Page);
var grid = new ej.grids.Grid({
    dataSource: purchaseData,
    allowPaging: true,
    allowTextWrap: true,
    toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
    editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true },
    created: function (args) {
        this.keyConfigs.enter = '';
    },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', type: 'number',
isPrimaryKey: true, validationRules: { required: true }, textAlign: 'Right',
width: 100 },
        { field: 'CustomerID', headerText: 'Customer ID', type: 'string',
width: 140 },
        { field: 'Freight', headerText: 'Freight', type: 'number', editType:
'numericedit', format: 'C2', textAlign: 'Right', width: 120 },
        { field: 'ShipAddress', headerText: 'Ship Address', type: 'string',
valueAccessor: valueAccessor, disableHtmlEncode: false, width: 180, edit: {
create: createShipAddressFn,
destroy: destroyShipAddressFn,
read: readShipAddressFn,
write: writeShipAddressFn }
}
}

```

```

    ],
    pageSettings: { pageSize: 7 },
    height: 255,
  });
  grid.appendTo('#Grid');

  function createShipAddressFn() {
    tbElem = document.createElement('textarea');
    return tbElem;
  }
  function destroyShipAddressFn() {
    textEditor.destroy();
  }
  function readShipAddressFn() {
    return textEditor.value;
  }
  function writeShipAddressFn(args) {
    textEditor = new ej.inputs.TextBox({
      multiline: true,
      value: args.rowData[args.column.field],
      floatLabelType: 'Auto',
    });
    textEditor.appendTo(tbElem);
  }

  function valueAccessor(field, data, column) {
    var value = data[field];
    if (value != undefined) {
      return value.split('\n').join('<br>');
    } else {
      return '';
    }
  }
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

[Render multiple columns in DropDownList component while editing](#)

Use the cell edit template feature of the Grid to render the DropDownList component in the Grid edit form.

The DropDownList has been provided with several options to customize each list item, group title, selected value, header, and footer element. By default, list items can be rendered as a single column in the DropDownList component. Instead of this, multiple columns can be rendered. This can be achieved by using the [headerTemplate](#) and [itemTemplate](#) properties of the DropDownList component.

This is demonstrated in the following sample.

INDEX.JS

```
var element;
var dropdownobj;
ej.grids.Grid.Inject(ej.grids.Edit, ej.grids.Toolbar);
var grid = new ej.grids.Grid({
    dataSource: data,
    toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
    editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100, isPrimaryKey: true },
        { field: 'CustomerID', headerText: 'Customer ID', width: 120, },
        { field: 'ShipCountry', width: 300, headerText: 'ShipCountry' ,
edit: {
            create: function () {
                element = document.createElement('input');
                return element;
            },
            read: function () {
                return dropdownobj.value;
            },
            destroy: function () {
                dropdownobj.destroy();
            },
            write: function (args) {
                dropdownobj = new ej.dropdowns.DropDownList({
                    dataSource: data,
                    value: args.rowData[args.column.field],
                    query: new ej.data.Query().select(['EmployeeID',
'ShipCountry']).take(10),
                    fields: { text: 'ShipCountry', value: 'ShipCountry' },
                    placeholder: 'Select a Country',
                    headerTemplate:
'<table><tr><th>EmployeeID</th><th>ShipCountry</th></tr></table>',
                    itemTemplate: '<div class="e-grid"><table class="e-
table"><tbody><tr><td class="e-rowcell">${EmployeeID}</td><td class="e-
rowcell">${ShipCountry}</td></tr> </tbody></table></div>'
                });
                dropdownobj.appendTo(element);
            }
        }
    ]
});
grid.appendTo('#Grid');
```

INDEX.HTML


```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <style>
    .content {
      margin: 0 auto;
      width: 550px;
    }
    table{
      width:100%;
      border-collapse: separate;
      table-layout: fixed;
    }
    th,td{
      border-width: 1px 0 0 1px;
      border-color: #e0e0e0;
      text-align: left;
      border-style: solid;
      display: table-cell;
    }
    th{
      line-height: 36px;
      text-indent: 16px;
    }
    .e-ddl-header{
      padding-right: 17px;
      border-width: 1px 0px 1px 0px;
      border-color: #e0e0e0;
      border-style: solid;

```

```

    }
    .e-dropdownbase .e-list-item{
        padding-right:0px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Access editor components

You can access the component instance from the component element using the `ej2_instances` property.

In the below demo, you can access the Editor component instance while adding or editing actions on the [actionComplete](#) event.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Edit, ej.grids.Toolbar);
var grid = new ej.grids.Grid({
    dataSource: data,
    actionComplete: function(args) {
        if (args.requestType === 'beginEdit' || args.requestType === 'add')
        {
            var tr = args.row;
            var numericTextBox = tr.querySelector('.e-numerictextbox'); //
            numeric TextBox component element
            if (numericTextBox) {
                console.log('NumericTextBox instance: ',
                (numericTextBox).ej2_instances[0]); // numeric TextBox instance
            }
            var dropDownList = tr.querySelector('.e-dropdownlist'); //
            dropDownList component element
            if (dropDownList) {
                console.log('DropDownList instance: ',
                (dropDownList).ej2_instances[0]); // dropDownList instance
            }
        }
    },
    toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],

```

```

editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true },
columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100, isPrimaryKey: true },
    { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
    { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
editType: 'numericedit', width: 120, format: 'C2' },
    { field: 'ShipCountry', headerText: 'Ship Country', editType:
'dropdownedit', width: 150 }
],
height: 273
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
<title>EJ2 Grid</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
.e-row[aria-selected="true"] .e-customizedExpandcell {
    background-color: #e0e0e0;
}
.e-grid.e-gridhover tr[role='row']:hover {
    background-color: #eee;
}

```

```

    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
}
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

In line editing in ##Platform_Name## Grid control

In Normal edit mode, when you start editing the currently selected record is changed to edit state. You can change the cell values and save edited data to the data source. To enable Normal edit, set the [editSettings.mode](#) as **Normal**.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Edit, ej.grids.Toolbar);
var grid = new ej.grids.Grid({
    dataSource: data,
    toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
    editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true, mode: 'Normal' },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100, isPrimaryKey: true },
        { field: 'CustomerID', headerText: 'Customer ID', width: 120, },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
editType: 'numericedit', width: 120, format: 'C2' },
        { field: 'ShipCountry', headerText: 'Ship Country', editType:
'dropdownedit', width: 150 }
    ],
    height: 273
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
      background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
      background-color: #eee;
    }
    .e-expand::before {
      content: '\e5b8';
    }
    .empImage {
      margin: 6px 16px;
      float: left;
      width: 50px;
      height: 50px;
    }
  </style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>

```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Normal edit mode is default mode of editing.

Automatically update the column based on another column edited value

You can update the column value based on another column edited value by using the Cell Edit Template feature.

In the below demo, we have update the **TotalCost** column value based on the **UnitPrice** and **UnitInStock** column value while editing.

INDEX.JS

```

var priceElem;
var priceObj;
var stockElem;
var stockObj;
ej.grids.Grid.Inject(ej.grids.Edit, ej.grids.Toolbar);
var grid = new ej.grids.Grid({
    dataSource: productData,
    editSettings: {
        allowEditing: true,
        allowAdding: true,
        allowDeleting: true,
        mode: 'Normal',
        newRowPosition: 'Top'
    },
    allowPaging: true,
    pageSettings: { pageCount: 5 },
    toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
    columns: [
        {
            field: 'ProductID',
            isPrimaryKey: true,
            headerText: 'Product ID',
            textAlign: 'Right',
            validationRules: { required: true, number: true },
            width: 140
        },
        {
            field: 'ProductName',

```

```

        headerText: 'Product Name',
        validationRules: { required: true },
        width: 140
    },
    {
        field: 'UnitPrice',
        headerText: 'UnitPrice',
        textAlign: 'Right',
        edit: {
            create: function() {
                priceElem = document.createElement('input');
                return priceElem;
            },
            read: function() {
                return priceObj.value;
            },
            destroy: function() {
                priceObj.destroy();
            },
            write: function(args) {
                priceObj = new ej.inputs.NumericTextBox({
                    value: args.rowData[args.column.field],
                    change: function(args) {
                        var formEle =
grid.element.querySelector('form').ej2_instances[0];
                        var totalCostFieldEle = formEle.getInputElement('TotalCost');
                        totalCostFieldEle.value = priceObj.value * stockObj.value;
                    }
                });
                priceObj.appendTo(priceElem);
            }
        },
        width: 140,
        format: 'C2',
        validationRules: { required: true }
    },
    {
        field: 'UnitsInStock',
        headerText: 'Units In Stock',
        textAlign: 'Right',
        edit: {
            create: function() {
                stockElem = document.createElement('input');
                return stockElem;
            },
            read: function() {
                return stockObj.value;
            },
            destroy: function() {
                stockObj.destroy();
            },
            write: function(args) {
                stockObj = new ej.inputs.NumericTextBox({
                    value: args.rowData[args.column.field],
                    change: function(args) {
                        var formEle =
grid.element.querySelector('form').ej2_instances[0];

```

```

        var totalCostFieldEle = formEle.getInputElement('TotalCost');
        totalCostFieldEle.value = priceObj.value * stockObj.value;
    }
    });
    stockObj.appendTo(stockElem);
}
},
width: 140,
validationRules: { required: true }
},
{
    field: 'TotalCost',
    headerText: 'Total Unit Cost',
    textAlign: 'Right',
    allowEditing: false,
    width: 140,
    format: 'C2',
}
]
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

```



```

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Cancel edit based on condition

You can prevent the CRUD operations of the Grid by using condition in the [actionBegin](#) event with requestType as **beginEdit** for editing, **add** for adding and **delete** for deleting actions.

In the below demo, we prevent the CRUD operation based on the **Role** column value. If the Role Column is **Employee**, we are unable to edit/delete that row.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Edit, ej.grids.Toolbar);
var isAddable = true;
var grid = new ej.grids.Grid({
    dataSource: data,
    toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
    editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true, mode: 'Normal' },
    columns: [

```

```

        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100, isPrimaryKey: true },
        { field: 'Role', headerText: 'Role', width: 120, },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
editType: 'numericedit', width: 120, format: 'C2' },
        { field: 'ShipCountry', headerText: 'Ship Country', editType:
'dropdownedit', width: 150 }
    ],
    actionBegin: actionBegin,
    height: 240
});
grid.appendTo('#Grid');
function actionBegin(args) {
    if (args.requestType == 'beginEdit') {
        if (args.rowData['Role'].toLowerCase() == 'employee') {
            args.cancel = true;
        }
    }
    if (args.requestType == 'delete') {
        if (args.data[0]['Role'].toLowerCase() == 'employee') {
            args.cancel = true;
        }
    }
    if (args.requestType == 'add') {
        if (!isAddable) {
            args.cancel = true;
        }
    }
}
}
var button = document.createElement('button');
button.innerText = 'Grid is Addable';
document.body.insertBefore(button, document.body.children[0])
button.addEventListener('click', btnClick.bind(this));
function btnClick(args) {
    args.target.innerText == 'Grid is Addable' ? (args.target.innerText =
'Grid is Not Addable') : (args.target.innerText = 'Grid is Addable');
    isAddable = !isAddable;
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Perform CRUD action programmatically

Grid methods can be used to perform CRUD operations programmatically. The [addRecord](#), [deleteRecord](#), and [startEdit](#) methods are used to perform CRUD operations in the following demo.

- To add a new record to the Grid, use the [addRecord](#) method. In this method, you can pass the data parameter to add a new record to the Grid, and the index parameter to add a record at a specific index. If you call this method with no parameters, it will create an empty row in the Grid.
- To change the selected row to the edit state, use the [startEdit](#) method.
- If you need to update the row data in the Grid's datasource, you can use the [updateRow](#) method. In this method, you need to pass the index value of the row to be updated along with the updated data.
- If you need to update the particular cell in the row, you can use the [setCellValue](#) method. In this method, you need to pass the primary key value of the data source, field name, and new value for the particular cell.
- To remove a selected row from the Grid, use the [deleteRecord](#) method. For both edit and delete operations, you must select a row first.

Note: In both normal and dialog editing modes, these methods can be used.

INDEX.JS

```
var grid = new ej.grids.Grid({
  dataSource: data,
  editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true},
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, isPrimaryKey:true },
    { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
    { field: 'ShipCity', headerText: 'Ship City', width: 150 },
    { field: 'ShipName', headerText: 'Ship Name', width: 150 }
  ],
  height: 230
});
grid.appendTo('#Grid');
document.getElementById('add').onclick = function () {
  grid.addRecord({ "OrderID": "10248", "CustomerID": "RTER", "ShipCity":
"America", "ShipName": "Hanari" });
};
document.getElementById('edit').onclick = function () {
  grid.startEdit();
};
document.getElementById('delete').onclick = function () {
  grid.deleteRecord();
};
document.getElementById('updaterow').onclick = function () {
  grid.updateRow(0, { OrderID: 10248, CustomerID: 'RTER', ShipCity:
'America', ShipName: 'Hanari' });
};
document.getElementById('updatecell').onclick = function () {
  grid.setCellValue(grid.currentViewData[0].OrderID, 'CustomerID', 'Value
Changed');
};
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <button id="edit">Edit</button>
    <button id="add">Add</button>
    <button id="delete">Delete</button>
    <button id="updaterow">Update Row</button>
    <button id="updatecell">UpdateCell</button>
    <div id="Grid"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {

```

```

    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Confirmation dialog

The delete confirm dialog can be shown when deleting a record by defining the [showDeleteConfirmDialog](#) as `true`

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Edit, ej.grids.Toolbar);
var grid = new ej.grids.Grid({
    dataSource: data,
    toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
    editSettings: { showDeleteConfirmDialog: true, allowEditing: true,
allowAdding: true, allowDeleting: true, mode: 'Normal' },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100, isPrimaryKey: true },
        { field: 'CustomerID', headerText: 'Customer ID', width: 120, },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
editType: 'numericedit', width: 120, format: 'C2' },
        { field: 'ShipCountry', headerText: 'Ship Country', editType:
'dropdownedit', width: 150 }
    ],
    height: 273
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The `showDeleteConfirmDialog` supports all type of edit modes.

Default column values on add new row

The grid provides an option to set the default value for the columns when adding a new record in it. To set a default value for the particular column by defining the [columns.defaultValue](#).

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Edit, ej.grids.Toolbar);
var grid = new ej.grids.Grid({
  dataSource: data,
  toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
  editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true, mode: 'Normal' },
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100, isPrimaryKey: true, validationRules: { required: true } },
    { field: 'CustomerID', headerText: 'Customer ID', width: 120,
validationRules: { required: true, minLength: 3 }, defaultValue: 'HANAR' },
    { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
editType: 'numericedit', width: 120, format: 'C2' },
    { field: 'ShipCountry', headerText: 'Ship Country', editType:
'dropdownedit', width: 150 }
  ],
  height: 273
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <style>

```



```

        .e-row[aria-selected="true"] .e-customizedExpandcell {
            background-color: #e0e0e0;
        }
        .e-grid.e-gridhover tr[role='row']:hover {
            background-color: #eee;
        }
        .e-expand::before {
            content: '\e5b8';
        }
        .empImage {
            margin: 6px 16px;
            float: left;
            width: 50px;
            height: 50px;
        }
    </style>
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
    </head>
    <body>

        <div id="container">
            <div id="Grid"></div>
        </div>
    <script>
    var ele = document.getElementById('container');
    if(ele) {
        ele.style.visibility = "visible";
    }
    </script>
    <script src="index.js" type="text/javascript"></script>
    </body></html>

```

Adding a new row at the bottom of the Grid

By default, a new row will be added at the top of the grid. You can change it by setting [editSettings.newRowPosition](#) as **Bottom**.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Edit, ej.grids.Toolbar);
var grid = new ej.grids.Grid({
    dataSource: data,
    toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
    editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
    true, newRowPosition: 'Bottom' },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
        width: 100, isPrimaryKey: true },
        { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
        width: 120, format: 'C2' },
        { field: 'ShipCountry', headerText: 'Ship Country', width: 150 }
    ],

```

```

        height: 273
    });
    grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <style>
        .e-row[aria-selected="true"] .e-customizedExpandcell {
            background-color: #e0e0e0;
        }
        .e-grid.e-gridhover tr[role='row']:hover {
            background-color: #eee;
        }
        .e-expand::before {
            content: '\e5b8';
        }
        .empImage {
            margin: 6px 16px;
            float: left;
            width: 50px;
            height: 50px;
        }
    </style>

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Add `newRowPosition` is supported for **Normal** and **Batch** editing modes.

Move the focus to a particular cell instead of first cell while editing a row

The [recordDoubleClick](#) event allows you to move the focus to the corresponding cell (the cell that you double-clicked to edit a row) instead of the first cell in edit form. With the help of this event, you can focus the double-clicked column in inline edit mode.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Page, ej.grids.Edit, ej.grids.Toolbar);
var grid = new ej.grids.Grid({
    dataSource: data,
    allowPaging: true,
    editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true,
    mode: "Normal"
    },
    recordDoubleClick: recordDoubleClick,
    actionComplete: actionComplete,
    columns: [
        { field: 'OrderID', isPrimaryKey: true, headerText: 'Order ID',
textAlign: 'Right', width: 120, type: 'number' },
        { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
        { field: 'Freight', headerText: 'Freight', editType: "numericedit",
textAlign: 'Right', width: 120, format: 'C2' },
        { field: 'OrderDate', headerText: 'Order Date', textAlign: 'Right',
width: 140, editType: "datetimepickeredit",
format: { type: "dateTime", format: "M/d/y hh:mm a" }, },
        { field: "ShipCountry", headerText: "Ship Country", editType:
"dropdownedit", width: 150, edit: { params: { popupHeight: "300px" } }
        }
    ],
    height: 220
});
grid.appendTo('#Grid');

```

```

var fieldName;
function recordDoubleClick(e) {
    var clickedColumnIndex = e.cell.getAttribute("data-colindex");
    fieldName = this.columnModel[parseInt(clickedColumnIndex)].field;
}
function actionComplete(e) {
    if (e.requestType === "beginEdit") {
        // focus the column
        e.form.elements[grid.element.getAttribute("id") + fieldName].focus();
    }
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```

```

    <div id="container">
        <div id="Grid"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Dialog editing in ##Platform_Name## Grid control

In Dialog edit mode, when you start editing the currently selected row data will be shown on a dialog. You can change the cell values and save edited data to the data source. To enable Dialog edit, set the [editSettings.mode](#) as **Dialog**.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Edit, ej.grids.Toolbar);
var grid = new ej.grids.Grid({
    dataSource: data,
    toolbar: ['Add', 'Edit', 'Delete'],
    editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true, mode: 'Dialog' },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100, isPrimaryKey: true },
        { field: 'CustomerID', headerText: 'Customer ID', width: 120, },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
editType: 'numericedit', width: 120, format: 'C2' },
        { field: 'ShipCountry', headerText: 'Ship Country', editType:
'dropdownedit', width: 150 }
    ],
    height: 273
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customize edit dialog

You can customize the Header and Footer appearance of the edit dialog in the [actionComplete](#) event based on `requestType` as `beginEdit` or `add`.

In the following example, We have customized the dialog's height and title of the dialog's header and also we have changed the button content name in the dialog's footer.

INDEX.JS

```
ej.grids.Grid.Inject(ej.grids.Edit, ej.grids.Toolbar);
var grid = new ej.grids.Grid({
  dataSource: data,
  toolbar: ['Add', 'Edit', 'Delete'],
  editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true, mode: 'Dialog' },
  actionComplete: function (args) {
    if ((args.requestType === 'beginEdit' || args.requestType ===
'add')) {
      let dialog = args.dialog;
      // set the height of the dialog
      dialog.height = 400;
      // change the header of the dialog
      dialog.header = args.requestType === 'beginEdit' ? 'Edit Record
of ' + args.rowData['CustomerID'] : 'New Customer';
      // change the footer button name of the dialog
      dialog.getButtons()[0].content = "Ok";
      dialog.getButtons()[1].content = "Close";
    }
  },
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100, isPrimaryKey: true },
    { field: 'CustomerID', headerText: 'Customer ID', width: 120, },
    { field: 'ShipCountry', headerText: 'Ship Country', width: 150 }
  ],
  height: 273
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```


The Grid add or edit dialog element has the max-height property, which is calculated based on the available window height. So, in the normal window (1920 x 1080), it is possible to set the dialog's height up to 658px.

Show or hide columns in dialog editing

The Grid has the option to show hidden columns or hide visible columns while editing in the dialog edit mode by using the [actionBegin](#) event of the Grid.

In the `actionBegin` event, when the `requestType` is `beginEdit` or `add`, the column will be shown or hidden using the `column.visible` property. When the `requestType` is `save`, the properties will be reset to their original state.

In the following example, the CustomerID column is rendered as a hidden column, and the ShipCountry column is rendered as a visible column. In the edit mode, the CustomerID column will be changed to a visible state and the ShipCountry column will be changed to a hidden state.

INDEX.JS

```
ej.grids.Grid.Inject(ej.grids.Toolbar, ej.grids.Edit);
var grid = new ej.grids.Grid({
  dataSource: data,
  toolbar: ['Add', 'Edit', 'Delete'],
  editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true, mode: 'Dialog' },
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100, isPrimaryKey: true },
    { field: 'CustomerID', headerText: 'Customer ID', width: 120,
visible: false },
    { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
editType: 'numericedit', width: 120, format: 'C2' },
    { field: 'ShipCountry', headerText: 'Ship Country', width: 150 }
  ],
  height: 265,
  actionBegin: function (args) {
    if ((args.requestType === 'beginEdit' || args.requestType ===
'add')) {
      for (var i = 0; i < this.columns.length; i++) {
        if (this.columns[i].field == "CustomerID") {
          this.columns[i].visible = true;
        }
        else if (this.columns[i].field == "ShipCountry") {
          this.columns[i].visible = false;
        }
      }
    }
    if (args.requestType === 'save') {
      for (var i = 0; i < this.columns.length; i++) {
        if (this.columns[i].field == "CustomerID") {
          this.columns[i].visible = false;
        }
        else if (this.columns[i].field == "ShipCountry") {
          this.columns[i].visible = true;
        }
      }
    }
  }
});
```

```

    }
  });
  grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
      background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
      background-color: #eee;
    }
    .e-expand::before {
      content: '\e5b8';
    }
    .empImage {
      margin: 6px 16px;
      float: left;
      width: 50px;
      height: 50px;
    }
  </style>

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Use wizard like dialog editing

Wizard helps you to create intuitive step-by-step forms to fill. You can achieve the wizard-like editing by using the dialog template feature. It supports your own editing template by defining the [editSettings.mode](#) as **Dialog** and [editSetting.template](#) as SCRIPT element ID or HTML string which holds the template.

The following example demonstrates the wizard-like editing in the grid with obtrusive validation.

INDEX.JS

```

var countryData = ej.data.DataUtil.distinct(data, 'ShipCountry', true);
ej.grids.Grid.Inject(ej.grids.Toolbar, ej.grids.Edit);
var grid = new ej.grids.Grid({
    dataSource: data,
    toolbar: ['Add', 'Edit', 'Delete'],
    editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true, mode: 'Dialog', template: '#dialogtemplate' },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100, isPrimaryKey: true },
        { field: 'CustomerID', headerText: 'Customer ID', width: 120, },
        { field: 'ShipCountry', headerText: 'Ship Country', width: 150 }
    ],
    height: 265,
    actionComplete: (args) => {
        if ((args.requestType === 'beginEdit' || args.requestType ===
'add')) {
            new ej.dropdowns.DropDownList({
                value: args.rowData.ShipCountry,
                popupHeight: '300px',
                floatLabelType: 'Always',
                dataSource: countryData,
                fields: {text: 'ShipCountry', value: 'ShipCountry'},
                placeholder: 'Ship Country'
            },
            args.form.elements.namedItem('ShipCountry'));
        }
    }
});

```

```

        new ej.buttons.CheckBox({
            label: 'Verified',
            checked: args.rowData.Verified
        }),
        args.form.elements.namedItem('Verified'));
    // Set initail Focus
    if (args.requestType === 'beginEdit') {
        args.form.elements.namedItem('CustomerID').focus();
    }
    initializeWizard();
}
});
grid.appendTo('#Grid');
function initializeWizard() {
    let currentTab = 0;
    document.getElementById('nextBtn').onclick = function () {
        if (validate()) {
            if (this.innerHTML !== 'SUBMIT') {
                currentTab++;
                nextpre(currentTab);
            } else {
                grid.endEdit();
            }
        }
    }
    function validate(tab) {
        let valid = true;
        [].slice.call(document.getElementById('tab' +
currentTab).querySelectorAll('[name]')).forEach(element => {
            element.form.ej2_instances[0].validate(element.name);
            if (element.getAttribute('aria-invalid') === 'true') {
                valid = false;
            }
        });
        if (!valid) {
            return false;
        }
        return true;
    }
    document.getElementById('prevBtn').onclick = function () {
        if (validate()) {
            currentTab--;
            nextpre(currentTab);
        }
    }
}
function nextpre(current) {
    let tabs = [].slice.call(document.getElementsByClassName('tab'))
    tabs.forEach(element => element.style.display = 'none');
    tabs[current].style.display = '';
    if(current) {
        document.getElementById('prevBtn').style.display = '';
        document.getElementById('nextBtn').innerHTML = 'SUBMIT';
    } else {
        document.getElementById('prevBtn').style.display = 'none';
        document.getElementById('nextBtn').innerHTML = 'NEXT';
    }
}

```

```
}
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.2/css/bootstrap.min.c
ss">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Grid"></div>
  </div>
  <script id="dialogtemplate" type="text/x-template">
    <div>
      <div id="tab0" class='tab'>
```

```

        <div class="form-row">
            <div class="form-group col-md-6">
                <div class="e-float-input e-control-wrapper">
                    <input id="OrderID" required name="OrderID"
type="text" value=${if(isAdd)} '' ${else} ${OrderID} ${/if} ${if(isAdd)} ''
${else} disabled ${/if} />
                    <span class="e-float-line"></span>
                    <label class="e-float-text e-label-top"
for="OrderID">Order ID</label>
                </div>
            </div>
        </div>
        <div class="form-row">
            <div class="form-group col-md-6">
                <div class="e-float-input e-control-wrapper">
                    <input id="CustomerID" required
name="CustomerID" type="text" value=${if(isAdd)} '' ${else} ${CustomerID}
${/if} />
                    <span class="e-float-line"></span>
                    <label class="e-float-text e-label-top"
for="CustomerID">Customer ID</label>
                </div>
            </div>
        </div>
        <div id=tab1 style="display: none" class='tab'>
            <div class="form-row">
                <div class="form-group col-md-6">
                    <input type="text" name="ShipCountry"
id="ShipCountry" value=${if(isAdd)} '' ${else} ${ShipCountry} ${/if} />
                </div>
            </div>
            <div class="form-row">
                <div class="form-group col-md-6">
                    <input type="checkbox" name="Verified" id="Verified"
${if(Verified)} checked ${/if} />
                </div>
            </div>
        </div>
        <div id='footer'>
            <button id="prevBtn" class="e-info e-btn" type="button"
style="display: none; float: left">Previous</button>

            <button id="nextBtn" class="e-info e-btn" type="button"
style="float: right">Next</button>
        </div>
    </script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customize Add/Edit Dialog footer

In dialog edit mode, a dialog will show up when editing the currently selected row or adding a new row. By default, you can save or cancel the edited changes by clicking the Save or Cancel button in the dialog's footer. Along with these buttons, it is possible to add a custom button in the footer section using the [actionComplete](#) event of the Grid.

In the following sample, using the `dialog` argument of the `actionComplete` event, the action for the custom button can be customized.

INDEX.JS

```
var grid = new ej.grids.Grid({
  dataSource: data,
  toolbar: ['Add', 'Edit', 'Delete'],
  editSettings: {
    allowEditing: true,
    allowAdding: true,
    allowDeleting: true,
    mode: 'Dialog',
  },
  actionComplete: actionComplete,
  columns: [{
    field: 'OrderID', headerText: 'Order ID', textAlign: 'Right', width:
100, isPrimaryKey: true },
    { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
    { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
editType: 'numericedit', width: 120, format: 'C2' },
    { field: 'ShipCountry', headerText: 'Ship Country', editType:
'dropdownedit', width: 150 },
  ]
});
grid.appendTo('#Grid');
function actionComplete(args) {
  if (args.requestType === 'beginEdit' || args.requestType === 'add') {
    var newFooterButton = {
      buttonModel: { content: 'custom' },
      click: onCustomButtonClick
    };
    args.dialog.buttons.push(newFooterButton);
    args.dialog.refresh();
  }
}
function onCustomButtonClick() {
  alert('Add/Edit dialog custom footer button clicked');
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
```

```

    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <style>
        .e-row[aria-selected="true"] .e-customizedExpandcell {
            background-color: #e0e0e0;
        }
        .e-grid.e-gridhover tr[role='row']:hover {
            background-color: #eee;
        }
        .e-expand::before {
            content: '\e5b8';
        }
        .empImage {
            margin: 6px 16px;
            float: left;
            width: 50px;
            height: 50px;
        }
    </style>
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
    <script>
var ele = document.getElementById('container');

```



```

if(ele) {
    ele.style.visibility = "visible";
}

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Template editing in ##Platform_Name## Grid control

Inline or dialog template editing

The dialog/inline template editing provides an option to customize the default behavior of dialog editing. Using the dialog template, you can render your own editors by defining the [editSettings.mode](#) as **Dialog/Inline** and [editSetting.template](#) as SCRIPT element ID or HTML string which holds the template.

In some cases, you need to add the new field editors in the dialog which are not present in the column model. In that situation, the dialog template will help you to customize the default edit dialog.

In the following sample, grid enabled with dialog template editing.

INDEX.JS

```

var countryData = ej.data.DataUtil.distinct(data, 'ShipCountry', true);
ej.grids.Grid.Inject(ej.grids.Edit, ej.grids.Toolbar);
var grid = new ej.grids.Grid({
    dataSource: data,
    toolbar: ['Add', 'Edit', 'Delete'],
    editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true, mode: 'Dialog', template: '#dialogtemplate' },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100, isPrimaryKey: true },
        { field: 'CustomerID', headerText: 'Customer ID', width: 120, },
        { field: 'ShipCountry', headerText: 'Ship Country', width: 150 }
    ],
    height: 265,
    actionBegin: function (args) {
        if (args.requestType === 'save') {
            // cast string to integer value.
            args.data['Freight'] =
parseFloat(args.form.querySelector("#Freight").value);
        }
    },
    actionComplete: function (args) {
        if ((args.requestType === 'beginEdit' || args.requestType ===
'add')) {
            // Add Validation Rules
            args.form.ej2_instances[0].addRules('Freight', {max: 500});
            // EJ2-control Rendering
            new ej.dropdowns.DropDownList({value: args.rowData.ShipCountry,
popupHeight: '200px', floatLabelType: 'Always',
dataSource: countryData, fields: {text: 'ShipCountry',
value: 'ShipCountry'}, placeholder: 'Ship Country'},
args.form.elements.namedItem('ShipCountry'));
            new ej.buttons.CheckBox({ label: 'Verified', checked:
args.rowData.Verified }, args.form.elements.namedItem('Verified'));

```

```

        new ej.inputs.NumericTextBox({value: args.rowData.Freight,
format: 'C2', placeholder: 'Freight', floatLabelType: 'Always' },
args.form.elements.namedItem('Freight'));
        // Set initail Focus
        if (args.requestType === 'beginEdit') {
            (args.form.elements.namedItem('CustomerID')).focus();
        }
    }
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.2/css/bootstrap.min.c
ss">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>

```

```

</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
    <script id="dialogtemplate" type="text/x-template">
        <div>
            <div class="form-row" >
                <div class="form-group col-md-6">
                    <div class="e-float-input e-control-wrapper">
                        <input id="OrderID" name="OrderID" type="text"
value=${if(isAdd)} '' ${else} ${OrderID} ${/if}  ${if(isAdd)} '' ${else}
disabled ${/if}/>
                        <span class="e-float-line"></span>
                        <label class="e-float-text e-label-top"
for="OrderID">Order ID</label>
                    </div>
                </div>
                <div class="form-group col-md-6">
                    <div class="e-float-input e-control-wrapper">
                        <input id="CustomerID" name="CustomerID" type="text"
value=${if(isAdd)} '' ${else} ${CustomerID} ${/if} />
                        <span class="e-float-line"></span>
                        <label class="e-float-text e-label-top"
for="CustomerID">Customer ID</label>
                    </div>
                </div>
            </div>
            <div class="form-row">
                <div class="form-group col-md-6">
                    <div>
                        <input id="ShipCountry" name="ShipCountry"
type="text" value=${if(isAdd)} '' ${else} ${ShipCountry} ${/if} />
                    </div>
                </div>
                <div class="form-group col-md-6">
                    <input id="Freight" value=${if(isAdd)} '' ${else}
${Freight} ${/if} />
                </div>
            </div>
            <div class="form-row">
                <div class="form-group col-md-12">
                    <div class="e-float-input e-control-wrapper">
                        <textarea type="text" name="ShipAddress"
id="ShipAddress" value=${if(isAdd)} '' ${else} ${ShipAddress}
${/if}>${if(isAdd)}  ${else} ${ShipAddress} ${/if}</textarea>
                        <span class="e-float-line"></span>
                        <label class="e-float-text e-label-top"
for="ShipAddress">Ship Address</label>
                    </div>
                </div>
            </div>
            <div class="form-row">
                <div class="form-group col-md-6">
                    <input type="checkbox" name="Verified" id="Verified"
${if(isAdd)} '' ${else} checked ${/if} />
                </div>
            </div>
        </div>
    </script>

```

```

        </div>
    </div>
</div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The template form editors should have **name** attribute.

Template context

The Essential JS2 packages has a built-in template engine. Using the template engine, you can access the row information inside the HTML element and bind the attributes, values, or elements based on this row information.

The following properties will be available at the time of template execution.

Property Name	Usage
---------------	-------

isAdd	A Boolean property; it defines whether the current row should be a new record or not.
-------	---

In the following code example, the **OrderID** textbox has been disabled by using the **isAdd** property.

,

// The disabled attributes will be added based on the isAdd property.

```

<input id="OrderID" name="OrderID" type="text" value=${if(isAdd)} " ${else} ${OrderID} ${/if}
${if(isAdd)}" ${else} disabled ${/if}/>

```

,

The following code example illustrates rendering the **OrderID** textbox, when a new record is added.

,

```

${if(isAdd)}
<div class="form-group col-md-6">
<div class="e-float-input e-control-wrapper">
<input id="OrderID" name="OrderID" type="text" value=${if(isAdd)} " ${else} ${OrderID} ${/if} />
<span class="e-float-line"></span>
<label class="e-float-text e-label-top" for="OrderID">Order ID</label>
</div>
</div>

```

`${/if}`

,

The dialog/inline dialog template syntax supports the ES6 expression string literals, and you can refer to the [Template Engine](#) for more template syntax.

Render editors as components

You can convert the form editors to EJ2 controls in the [actionComplete](#) event based on the `requestType` as `beginEdit` or `add`.

The following code example illustrates rendering the drop-down list control in the `actionComplete` event.

``ts`

```
actionComplete: (args: DialogEditEventArgs) => {
  if ((args.requestType === 'beginEdit' || args.requestType === 'add')) {
    let countryData: {}[] = DataUtil.distinct(data, 'ShipCountry', true);
    new DropDownList({value: args.rowData.ShipCountry, popupHeight: '200px', floatLabelType: 'Always',
      dataSource: countryData, fields: {text: 'ShipCountry', value: 'ShipCountry'}, placeholder: 'Ship Country'},
      args.form.elements.namedItem('ShipCountry') as HTMLInputElement);
  }
}
```

}

,

Get value from editor

You can read, format, and update the current editor value in the [actionBegin](#) event at the time of setting `requestType` to `save`.

In the following code example, the `freight` value has been formatted and updated.

``ts`

```
actionBegin: (args: SaveEventArgs) => {
  if (args.requestType === 'save') {
    // cast string to integer value.
    args.data['Freight'] = parseFloat(args.form.querySelector("#Freight").value);
  }
}
```

}

,

Set focus to editor

By default, the first input element in the dialog will be focused while opening the dialog. If the first input element is in disabled or hidden state, focus the valid input element in the [actionComplete](#) event based on `requestType` as `beginEdit`.

``ts`

```

actionComplete: (args: DialogEditEventArgs) => {
// Set initail Focus
if (args.requestType === 'beginEdit') {
(args.form.elements.namedItem('CustomerID')as HTMLInputElement).focus();
}
}
,

```

Adding validation rules for custom editors

If you have used additional fields that are not present in the column model, then add the validation rules to the [actionComplete](#) event.

```

`ts
actionComplete: (args: DialogEditEventArgs) => {
if ((args.requestType === 'beginEdit' || args.requestType === 'add')) {
// Add Validation Rules
args.form.ej2_instances[0].addRules('Freight', {max: 500});
}
}
,

```

Render tab component inside the dialog template

You can use the [tab](#) component inside the dialog edit UI using the dialog template feature. The dialog template feature can be enabled by defining the [editSettings.mode](#) as **Dialog** and [editSetting.template](#) as SCRIPT element ID or HTML string which holds the template.

To include the tab components in the dialog, follow the given steps:

Step 1:

Initialize the template for your tab component.

```

,

<div>
<div id="edittab"></div>
<div id="tab1">
<div class="form-row" >
<div class="form-group col-md-6">
<div class="e-float-input e-control-wrapper">
<input id="OrderID" name="OrderID" type="text" value=${if(isAdd)} " ${else} ${OrderID} ${/if}
${if(isAdd)} " ${else} disabled ${/if} />
<span class="e-float-line"></span>

```

```

<label class="e-float-text e-label-top" for="OrderID">Order ID</label>
</div>
</div>
</div>
<div class="form-row" >
<div class="form-group col-md-6">
<div class="e-float-input e-control-wrapper">
<input id="CustomerID" name="CustomerID" type="text" value=${if(isAdd)} " ${else} ${CustomerID}
${/if} />
<span class="e-float-line"></span>
<label class="e-float-text e-label-top" for="CustomerID">Customer ID</label>
</div>
</div>
</div>
<button id='next' class='e-info e-btn' style="float: right" type='button'> next</button>
</div>
<div id="tab2" style='display: none'>
<div class="form-row" >
<div class="form-group col-md-6">
<input type="text" name="ShipCountry" id="ShipCountry" value=${if(isAdd)} " ${else} ${ShipCountry}
${/if} />
</div>
</div>
<div class="form-row">
<div class="form-group col-md-6">
<input type="checkbox" name="Verified" id="Verified" ${if(isAdd)} " ${else} checked ${/if} />
</div>
</div>
<button id='submit' class='e-info e-btn' style="float: right" type='button'> submit</button>
</div>
</div>
,

```

Step 2:

To render the tab component, use the [actionComplete](#) event of the grid.

```

`ts
let tabObj: Tab = new Tab({
showCloseButton: false,
selecting: (e) => {if(e.isSwiped) {e.cancel = true;} if(e.selectingIndex === 1) {e.cancel = !validate(1)}},
items: [
{ header: { 'text': 'Details' }, content: '#tab1' },
{ header: { 'text': 'Verify' }, content: '#tab2' },
]
});
tabObj.appendTo('#edittab');
`

```

The following example demonstrates rendering the tab control inside the edit dialog. The tab control contains two tabs. You can fill the first tab and then navigate to the second tab. The validation for first tab will be done before navigating to the second tab.

INDEX.JS

```

var countryData = ej.data.DataUtil.distinct(data, 'ShipCountry', true);
var grid = new ej.grids.Grid({
  dataSource: data,
  toolbar: ['Add', 'Edit', 'Delete'],
  editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true, mode: 'Dialog', template: '#dialogtemplate' },
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100, isPrimaryKey: true, validationRules: { required: true } },
    { field: 'CustomerID', headerText: 'Customer ID', width: 120,
validationRules: { required: true } },
    { field: 'ShipCountry', headerText: 'Ship Country', width: 150 }
  ],
  height: 265,
  actionComplete: (args) => {
    if ((args.requestType === 'beginEdit' || args.requestType === 'add'))
    {
      var tabObj = new ej.navigations.Tab({
        showCloseButton: false,
        selecting: (e) => {if(e.isSwiped) {e.cancel = true;}
if(e.selectingIndex === 1) {e.cancel = !validate(1)}},
        items: [
          { header: { 'text': 'Details' }, content: '#tab1' },
          { header: { 'text': 'Verify' }, content: '#tab2' },
        ]
      });
      tabObj.appendTo('#edittab');
      new ej.dropdowns.DropDownList({value: args.rowData.ShipCountry,
popupHeight: '200px', floatLabelType: 'Always',
        dataSource: countryData, fields: {text: 'ShipCountry', value:
'ShipCountry'}, placeholder: 'Ship Country'},
args.form.elements.namedItem('ShipCountry'));
    }
  }
});

```



```

        new ej.buttons.CheckBox({ label: 'Verified', checked:
args.rowData.Verified }, args.form.elements.namedItem('Verified'));
        // Set initail Focus
        if (args.requestType === 'beginEdit') {
            (args.form.elements.namedItem('CustomerID')).focus();
        }
        document.getElementById('next').onclick = () => {
            moveNext();
        }
        function validate(tab) {
            let valid = true;
            [].slice.call(document.getElementById('tab' +
tab).querySelectorAll('[name]')).forEach(element => {
                element.form.ej2_instances[0].validate(element.name);
                if (element.getAttribute('aria-invalid') === 'true') {
                    valid = false;
                }
            });
            if (!valid) {
                return false;
            }
            return true;
        }
        function moveNext() {
            if (validate(1)) {
                tabObj.select(1);
            }
        }
        document.getElementById('submit').onclick = () => {
            if (validate(2)) {
                grid.endEdit();
            }
        }
    }
}
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.2/css/bootstrap.min.c
ss">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
    <script id='dialogtemplate' type="text/x-template">
        <div>
            <div id="edittab"></div>
            <div id="tab1">
                <div class="form-row" >
                    <div class="form-group col-md-6">
                        <div class="e-float-input e-control-wrapper">
                            <input id="OrderID" name="OrderID" type="text"
value=${if(isAdd)} '' ${else} ${OrderID} ${/if} ${if(isAdd)} '' ${else}
disabled ${/if} />
                            <span class="e-float-line"></span>
                            <label class="e-float-text e-label-top"
for="OrderID">Order ID</label>
                        </div>
                    </div>
                </div>
                <div class="form-row" >
                    <div class="form-group col-md-6">
                        <div class="e-float-input e-control-wrapper">
                            <input id="CustomerID" name="CustomerID"
type="text" value=${if(isAdd)} '' ${else} ${CustomerID} ${/if} />
                            <span class="e-float-line"></span>
                            <label class="e-float-text e-label-top"
for="CustomerID">Customer ID</label>
                        </div>
                    </div>
                </div>
                <button id='next' class='e-info e-btn' style="float: right"
type='button'> next</button>
            </div>
        </div>
    </script>

```

```

        <div id="tab2" style='display: none'>
            <div class="form-row" >
                <div class="form-group col-md-6">
                    <input type="text" name="ShipCountry"
id="ShipCountry" value=${if(isAdd)} '' ${else} ${ShipCountry} ${if} />
                </div>
            </div>
            <div class="form-row">
                <div class="form-group col-md-6">
                    <input type="checkbox" name="Verified"
id="Verified" ${if(isAdd)} '' ${else} checked ${if} />
                </div>
            </div>
            <button id='submit' class='e-info e-btn' style="float:
right" type='button'> submit</button>
        </div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Batch editing in ##Platform_Name## Grid control

In Batch edit mode, when you double-click on the grid cell, then the target cell changed to edit state. You can bulk save (added, changed and deleted data in the single request) to data source by click on the toolbar's **Update** button or by externally invoking the `[batchSave]../api/grid/edit/#batchsave` method. To enable Batch edit, set the [editSettings.mode](#) as **Batch**.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Edit, ej.grids.Toolbar);
var grid = new ej.grids.Grid({
    dataSource: data,
    toolbar: ['Add', 'Delete', 'Update', 'Cancel'],
    editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true, mode: 'Batch' },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100, isPrimaryKey: true },
        { field: 'CustomerID', headerText: 'Customer ID', width: 120, },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
editType: 'numericedit', width: 120, format: 'C2' },
        { field: 'ShipCountry', headerText: 'Ship Country', editType:
'dropdownedit', width: 150 }
    ],
    height: 273
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
      background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
      background-color: #eee;
    }
    .e-expand::before {
      content: '\e5b8';
    }
    .empImage {
      margin: 6px 16px;
      float: left;
      width: 50px;
      height: 50px;
    }
  </style>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>

```

```

</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

If a column's [allowEditing](#) property is set to false, then the focus can be skipped in that non-editable column by clicking the tab or shift-tab key while in batch edit mode.

Automatically update the column based on another column edited value in batch mode

You can update the column value based on another column edited value in Batch mode by using the Cell Edit Template feature.

In the below demo, we have update the **TotalCost** column value based on the **UnitPrice** and **UnitInStock** column value while editing.

INDEX.JS

```

var priceElem;
var priceObj;
var stockElem;
var stockObj;
ej.grids.Grid.Inject(ej.grids.Edit, ej.grids.Toolbar);
var grid = new ej.grids.Grid({
    dataSource: productData,
    editSettings: {
        allowEditing: true,
        allowAdding: true,
        allowDeleting: true,
        mode: 'Batch',
        newRowPosition: 'Top'
    },
    allowPaging: true,
    pageSettings: { pageCount: 5 },
    toolbar: ['Add', 'Delete', 'Update', 'Cancel'],
    columns: [
        {
            field: 'ProductID',
            isPrimaryKey: true,
            headerText: 'Product ID',
            textAlign: 'Right',
            validationRules: { required: true, number: true },
            width: 140
        },
        {
            field: 'ProductName',
            headerText: 'Product Name',

```

```

validationRules: { required: true },
width: 140
},
{
  field: 'UnitPrice',
  headerText: 'UnitPrice',
  textAlign: 'Right',
  edit: {
    create: function() {
      priceElem = document.createElement('input');
      return priceElem;
    },
    read: function() {
      return priceObj.value;
    },
    destroy: function() {
      priceObj.destroy();
    },
    write: function(args) {
      var rowData = args.rowData;
      var rowIndex = grid.getRowInfo(args.row).rowIndex;
      priceObj = new ej.inputs.NumericTextBox({
        value: args.rowData[args.column.field],
        change: function(args) {
          var totalCostValue = args.value * rowData['UnitsInStock'];
          grid.updateCell(rowIndex, 'TotalCost', totalCostValue);
        }
      });
      priceObj.appendTo(priceElem);
    }
  },
  width: 140,
  format: 'C2',
  validationRules: { required: true }
},
{
  field: 'UnitsInStock',
  headerText: 'Units In Stock',
  textAlign: 'Right',
  edit: {
    create: function() {
      stockElem = document.createElement('input');
      return stockElem;
    },
    read: function() {
      return stockObj.value;
    },
    destroy: function() {
      stockObj.destroy();
    },
    write: function(args) {
      var rowData = args.rowData;
      var rowIndex = grid.getRowInfo(args.row).rowIndex;
      stockObj = new ej.inputs.NumericTextBox({
        value: args.rowData[args.column.field],
        change: function(args) {
          var totalCostValue = args.value * rowData['UnitPrice'];

```

```

        grid.updateCell(rowIndex, 'TotalCost', totalCostValue);
    }
    });
    stockObj.appendTo(stockElem);
}
},
width: 140,
validationRules: { required: true }
},
{
    field: 'TotalCost',
    headerText: 'Total Unit Cost',
    textAlign: 'Right',
    width: 140,
    format: 'C2',
}
]
});
grid.appendTo('#Grid');
grid.cellEdit= function(args){
    if(args.columnName == "TotalCost"){
        args.cancel= true;
    }
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Cancel edit based on condition in batch mode

You can prevent the CRUD operations of the Batch edit Grid by using condition in the [cellEdit](#), [beforeBatchAdd](#) and [beforeBatchDelete](#) events for Edit, Add and Delete actions respectively.

In the below demo, we prevent the CRUD operation based on the **Role** column value. If the Role Column is **Employee**, we are unable to edit/delete that row.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Edit, ej.grids.Toolbar);
var isAddable = true;
var grid = new ej.grids.Grid({
    dataSource: data,
    toolbar: ['Add', 'Delete', 'Update', 'Cancel'],

```



```

        editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true, mode: 'Batch' },
        columns: [
            { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100, isPrimaryKey: true },
            { field: 'Role', headerText: 'Role', width: 120, },
            { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
editType: 'numericedit', width: 120, format: 'C2' },
            { field: 'ShipCountry', headerText: 'Ship Country', editType:
'dropdownedit', width: 150 }
        ],
        cellEdit: cellEdit,
        beforeBatchAdd: beforeBatchAdd,
        beforeBatchDelete: beforeBatchDelete,
        height: 240
    });
grid.appendTo('#Grid');
function cellEdit(args) {
    if (args.rowData['Role'] == 'Employee') {
        args.cancel = true;
    }
}
function beforeBatchAdd(args) {
    if (!isAddable) {
        args.cancel = true;
    }
}
function beforeBatchDelete(args) {
    if (args.rowData['Role'] == 'Employee') {
        args.cancel = true;
    }
}
var button = document.createElement('button');
button.innerText = 'Grid is Addable';
document.body.insertBefore(button, document.body.children[0]);
button.addEventListener('click', btnClick.bind(this));
function btnClick(args) {
    args.target.innerText == 'Grid is Addable' ? (args.target.innerText =
'Grid is Not Addable') : (args.target.innerText = 'Grid is Addable');
    isAddable = !isAddable;
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>

```

```
</body></html>
```

Confirmation dialog

By default, grid will show the confirm dialog when saving or canceling or performing any actions like filtering, sorting, etc.

INDEX.JS

```
ej.grids.Grid.Inject(ej.grids.Edit, ej.grids.Toolbar);
var grid = new ej.grids.Grid({
  dataSource: data,
  toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
  editSettings: { showConfirmDialog: true, showDeleteConfirmDialog: true,
allowEditing: true, allowAdding: true, allowDeleting: true, mode: 'Batch' },
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100, isPrimaryKey: true },
    { field: 'CustomerID', headerText: 'Customer ID', width: 120, },
    { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
editType: 'numericedit', width: 120, format: 'C2' },
    { field: 'ShipCountry', headerText: 'Ship Country', editType:
'dropdownedit', width: 150 }
  ],
  height: 273
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

* [editSettings.showConfirmDialog](#) requires the `editSettings.mode` to be `Batch`

* If [editSettings.showConfirmDialog](#) set to `false`, then confirmation dialog does not display in batch editing.

How to make editing in single click and arrow keys

When using batch mode, the TAB key allows you to edit or move to the next cell or row from the current record by default. Using the grid's load event, the same functionality can also be achieved by pressing the arrow keys. Additionally, the `editCell` method of the grid allows for cells to be made editable with a single click.

In the following sample, the [load](#) event of the Grid will be used to bind the keydown event handler. When any arrow key is pressed, the `editCell` method of the Grid will be used to identify the next or previous cell (td) and set it to be editable. Additionally, it is possible to enable editing of a cell with a single click by utilizing the `editCell` method within the [created](#) event of the Grid.

INDEX.JS

```
var grid = new ej.grids.Grid({
  dataSource: data,
  allowPaging: true,
  enableHover: false,
  created: created,
  load: load,
  editSettings: {
    allowEditing: true,
    allowAdding: true,
    allowDeleting: true,
    mode: 'Batch',
  },
  toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100, isPrimaryKey: true,
      visible: false, validationRules: { required: true, number: true }
    },
    { field: 'CustomerID', headerText: 'Customer ID', width: 120,
validationRules: { required: true }
    },
    { field: 'Freight', headerText: 'Freight', format: 'C2', width: 100,
textAlign: 'Right' },
    { field: 'OrderDate', headerText: 'Order Date', editType:
'datepickeredit', format: 'yMd', width: 100},
    { field: 'ShipCountry', headerText: 'Ship Country', width: 100 },
  ],
});
grid.appendTo('#Grid');
function created(args) {
  grid.getContentTable().addEventListener('click', function (args) {
    if (args.target.classList.contains('e-rowcell')) {

grid.editModule.editCell(parseInt(args.target.getAttribute('index')),
      grid.getColumnByIndex(parseInt(args.target.getAttribute('data-
colindex'))).field);
    }
  });
}
function load() {
  grid.element.addEventListener('keydown', function (e) {
    var closesttd = e.target.closest('td');
    if (e.keyCode === 39 && !isNullOrUndefined(closesttd.nextSibling)) {
      editACell(closesttd.nextSibling);
    }
    if (e.keyCode === 37 &&
!isNullOrUndefined(closesttd.previousSibling) &&
      !grid.getColumnByIndex(
```

```

        parseInt(closesttd.previousSibling.getAttribute('data-
colindex'))).isPrimaryKey)
    {
        editACell(closesttd.previousSibling);
    }
    if (e.keyCode === 40 &&
!isNullOrUndefined(closesttd.closest('tr').nextSibling)) {
        editACell(
            closesttd.closest('tr').nextSibling.querySelector('td')[
                parseInt(closesttd.getAttribute('data-colindex'))]);
    }
    if (e.keyCode === 38 &&
!isNullOrUndefined(closesttd.closest('tr').previousSibling)) {
        editACell(
            closesttd.closest('tr').previousSibling.querySelector('td')[
                parseInt(closesttd.getAttribute('data-colindex'))]);
    }
});
}
function editACell(args) {
    grid.editModule.editCell(
        parseInt(args.getAttribute('index')),
        grid.getColumnByIndex(parseInt(args.getAttribute('data-
colindex'))).field);
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Command column editing in ##Platform_Name## Grid control

The **command** column provides an option to add CRUD action buttons in a column. This can be defined by the [column.commands](#) property.

The available built-in command buttons are:

Command Button	Actions
----- -----	
Edit	Edit the current row.
Delete	Delete the current row.

| Save | Update the edited row. |

| Cancel | Cancel the edited state. |

INDEX.JS

```
ej.grids.Grid.Inject(ej.grids.CommandColumn, ej.grids.Edit);
var grid = new ej.grids.Grid({
  dataSource: employeeData,
  editSettings: { allowEditing: true, allowDeleting: true },
  columns: [
    { field: 'EmployeeID', isPrimaryKey: 'true', headerText: 'Employee ID', textAlign: 'Right', width: 125 },
    { field: 'FirstName', headerText: 'Name', width: 120 },
    { field: 'Title', headerText: 'Title', width: 170 },
    { headerText: 'Commands', width: 120, commands: [{ type: 'Edit', buttonOption: { cssClass: 'e-flat', iconCss: 'e-edit e-icons' } }, { type: 'Delete', buttonOption: { cssClass: 'e-flat', iconCss: 'e-delete e-icons' } }, { type: 'Save', buttonOption: { cssClass: 'e-flat', iconCss: 'e-update e-icons' } }, { type: 'Cancel', buttonOption: { cssClass: 'e-flat', iconCss: 'e-cancel-icon e-icons' } } ] }
  ],
  height: 310
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-calendars/styles/material.css" rel="stylesheet">
```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-grids/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Custom command column

The custom command buttons can be added in a column by using the [column.commands](#) property and the action for the custom buttons can be defined in the [commandClick](#) event.

INDEX.JS

```

var commandClick = function(args){
    alert(JSON.stringify(args.rowData));
}
ej.grids.Grid.Inject(ej.grids.CommandColumn);
var grid = new ej.grids.Grid({
    dataSource: employeeData,
    commandClick:commandClick,
    columns: [
        { field: 'EmployeeID', headerText: 'Employee ID', textAlign:
'Right', width: 125 },
        { field: 'FirstName', headerText: 'Name', width: 120 },
        { field: 'Title', headerText: 'Title', width: 170 },
        { headerText: 'Commands', width: 120, commands: [{ buttonOption: {
content: 'Details', cssClass: 'e-flat' } } ] },
    ],
    height: 315
});
grid.appendTo('#Grid');

```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Grid"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Validation in ##Platform_Name## Grid control

Column validation

Column validation allows you to validate the edited or added row data and it display errors for invalid fields before saving data.

Grid uses **Form Validator** component for column validation. You can set validation rules by defining the [columns.validationRules](#).

INDEX.JS

```
ej.grids.Grid.Inject(ej.grids.Edit, ej.grids.Toolbar);
var grid = new ej.grids.Grid({
  dataSource: data,
  toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
  editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true, mode: 'Normal' },
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100, isPrimaryKey: true, validationRules: { required: true } },
    { field: 'CustomerID', headerText: 'Customer ID', width: 120,
validationRules: { required: true, minLength: 3 } },
    { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
editType: 'numericedit', width: 120, format: 'C2' },
    { field: 'ShipCountry', headerText: 'Ship Country', editType:
'dropdownedit', width: 150 }
  ],
  height: 273
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Custom validation

You can define your own custom validation rules for the specific columns by using **Form Validator custom rules**.

In the below demo, custom validation applied for **CustomerID** column.

INDEX.JS

```

var customFn = function(args) {

```

```

        return args['value'].length >= 5;
    };
    ej.grids.Grid.Inject(ej.grids.Edit, ej.grids.Toolbar);
    var grid = new ej.grids.Grid({
        dataSource: data,
        toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
        editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true, mode: 'Normal' },
        columns: [
            { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100, isPrimaryKey: true, validationRules: { required: true } },
            { field: 'CustomerID', headerText: 'Customer ID', width: 120,
validationRules: { required: true, minLength: [customFn, 'Need at least 5
letters'] } },
            { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
editType: 'numericedit', width: 120, format: 'C2' },
            { field: 'ShipCountry', headerText: 'Ship Country', editType:
'dropdownedit', width: 150 }
        ],
        height: 273
    });
    grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

```

```

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Custom validation based on dropdown change

You can apply validation rules and messages to a column based on another column value in edit mode. You can achieve this requirement by using the custom validation feature of Grid.

In the following sample, dropdownlist edit type is used for the **Role** and **Salary** columns. Here, you can apply the custom validation in the **Salary** column based on the value selected in the **Role** column.

INDEX.JS

```

var jobRole = [
    { role: 'TeamLead', destId: '0' },
    { role: 'Manager', destId: '1' },
    { role: 'Engineer', destId: '2' },
    { role: 'Sales', destId: '3' },
    { role: 'Support', destId: '4' },
];
var salaryDetails= [

```

```

    { salary: '11000', destId: '1' },
    { salary: '13500', destId: '2' },
    { salary: '16500', destId: '2' },
    { salary: '18500', destId: '1' },
    { salary: '21500', destId: '2' },
    { salary: '23000', destId: '2' },
  ];
  var grid = new ej.grids.Grid({
    dataSource: employeeDetails,
    allowPaging: true,
    toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
    editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true },
    load: () => {
      var column = grid.getColumnByField('Salary');
      column.validationRules = {
        required: [customFn, window['Please enter valid salary']],
      };
    },
    columns: [
      {
        field: 'EmployeeID',
        headerText: 'Employee ID',
        isPrimaryKey: true,
        textAlign: 'Right',
        width: 120
      },
      {
        field: 'Role',
        headerText: 'Role',
        width: 120,
        editType: 'dropdownedit',
        edit: {
          params: {
            query: new ej.data.Query(),
            dataSource: jobRole,
            fields: { value: 'role', text: 'role' },
            allowFiltering: true,
            change: ValChange
          }
        }
      },
      {
        field: 'Salary',
        headerText: 'Salary',
        width: 160,
        editType: 'dropdownedit',
        edit: {
          params: {
            query: new ej.data.Query(),
            dataSource: salaryDetails,
            fields: { value: 'salary', text: 'salary' },
            allowFiltering: true,
          }
        }
      }
    ],
  });

```

```

        field: 'Address',
        headerText: 'Address',
        width: 160
    }
    ]
});
grid.appendTo('#Grid');
window.role = '';
function customFn(args) {
    switch (window['role']) {
        case 'Engineer':
            if (args.value > 10000 && args.value < 15000) {
                return true;
            } else {
                this.rules['Salary']['required'][1] = 'Please enter valid
Engineer Salary';
            }
            break;
        case 'TeamLead':
            if (args.value > 15000 && args.value < 20000) {
                return true;
            } else {
                this.rules['Salary']['required'][1] = 'Please enter valid
TeamLead Salary';
            }
            break;
        case 'Manager':
            if (args.value > 20000 && args.value < 25000) {
                return true;
            } else {
                this.rules['Salary']['required'][1] = 'Please enter valid
Manager Salary';
            }
            break;
        case 'Sales':
            if (args.value > 5000 && args.value < 25000) {
                return true;
            } else {
                this.rules['Salary']['required'][1] = 'Please enter valid
Manager Salary';
            }
            break;
        case 'Support':
            if (args.value > 10000 && args.value < 19000) {
                return true;
            } else {
                this.rules['Salary']['required'][1] = 'Please enter valid
Manager Salary';
            }
            break;
    }
    return false;
}
function ValChange(args) {
    window['role'] = args.value;
}

```


INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
      background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
      background-color: #eee;
    }
    .e-expand::before {
      content: '\e5b8';
    }
    .empImage {
      margin: 6px 16px;
      float: left;
      width: 50px;
      height: 50px;
    }
  </style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>

```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Persisting data in server in ##Platform_Name## Grid control

Edited data can be persisted in the database using the RESTful web services.

All the CRUD operations in the grid are done through [DataManager](#). The **DataManager** has an option to bind all the CRUD related data in server-side.

For your information, the ODataAdaptor persists data in the server as per OData protocol.

In the below section, we have explained how to get the edited data details on the server-side using the [UrlAdaptor](#).

Using URL adaptor

You can use the [UrlAdaptor](#) of **DataManager** when binding data source from remote data. In the initial load of grid, data are fetched from remote data and bound to the grid using `url` property of **DataManager**. You can map The CRUD operation in grid can be mapped to server-side Controller actions using the properties `insertUrl`, `removeUrl`, `updateUrl`, `crudUrl` and `batchUrl`.

The following code example describes the above behavior.

```

`ts
import { Grid, Edit, Toolbar } from '@syncfusion/ej2-grids';
import { DataManager, UrlAdaptor } from '@syncfusion/ej2-data';
Grid.Inject(Edit, Toolbar);
let data: DataManager = new DataManager({
    url: "Home/DataSource",
    updateUrl: "Home/Update",
    insertUrl: "Home/Insert",
    removeUrl: "Home/Delete",
    adaptor: new UrlAdaptor
});
let grid: Grid = new Grid({

```

```

dataSource: data,
toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
editSettings: { allowEditing: true, allowAdding: true, allowDeleting: true, mode: 'Normal' },
columns: [
{ field: 'OrderID', headerText: 'Order ID', textAlign: 'Right', width: 100, isPrimaryKey: true,
validationRules: { required: true } },
{ field: 'CustomerID', headerText: 'Customer ID', width: 120, validationRules: { required: true, minLength:
3 }, defaultValue: 'HANAR' },
{ field: 'Freight', headerText: 'Freight', textAlign: 'Right', editType: 'numericedit', width: 120, format: 'C2'
},
{ field: 'ShipCountry', headerText: 'Ship Country', editType: 'dropdownedit', width: 150 }
],
height: 265
});
grid.appendTo('#Grid');
`

```

Also, when using the `UrlAdaptor`, you need to return the data as JSON from the controller action and the JSON object must contain a property as `result` with `dataSource` as its value and one more property `count` with the `dataSource` total records count as its value.

The following code example describes the above behavior.

```

`ts
public ActionResult DataSource(DataManager dm)
{
var DataSource = OrderRepository.GetAllRecords();
DataResult result = new DataResult();
result.result = DataSource.Skip(dm.Skip).Take(dm.Take).ToList();
result.count = result.result.Count;
return Json(result, JsonRequestBehavior.AllowGet);
}
public class DataResult
{
public List<EditableOrder> result { get; set; }
public int count { get; set; }
}
`

```

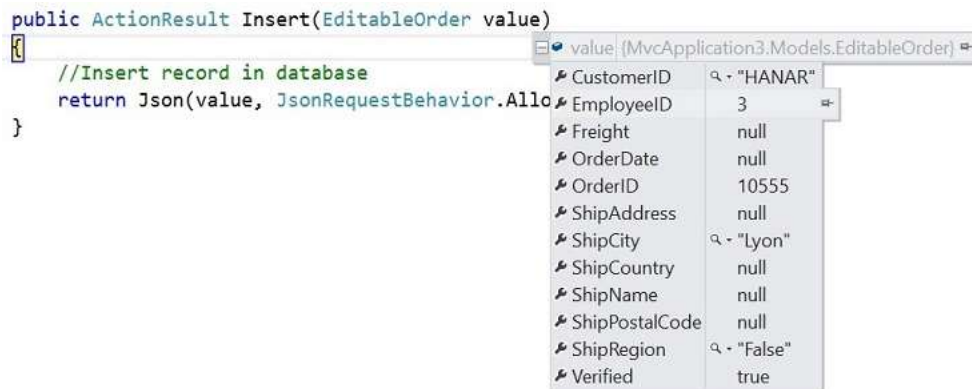
Insert record

Using the `insertUrl` property, you can specify the controller action mapping URL to perform insert operation on the server-side.

The following code example describes the above behavior.

```
`ts
public ActionResult Insert(EditableOrder value)
{
    //Insert record in database
}
`
```

The newly added record details are bound to the `value` parameter. Please refer to the following screenshot.



Update record

Using the `updateUrl` property, the controller action mapping URL can be specified to perform save/update operation on the server-side.

The following code example describes the previous behavior.

```
`ts
public ActionResult Update(EditableOrder value)
{
    //Update record in database
}
`
```

The updated record details are bound to the `value` parameter. Please refer to the following screenshot.



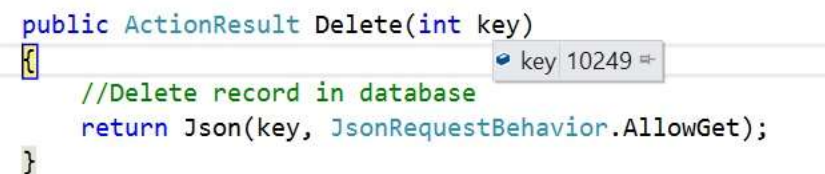
Delete record

Using the `removeUrl` property, the controller action mapping URL can be specified to perform delete operation on the server-side.

The following code example describes the previous behavior.

```
`ts
public ActionResult Delete(int key)
{
    //Delete record in database
}
```

The deleted record primary key value is bound to the `key` parameter. Please refer to the following screenshot.



CRUD URL

Using the `crudUrl` property, the controller action mapping URL can be specified to perform all the CRUD operation at server-side using a single method instead of specifying separate controller action method for CRUD (insert, update and delete) operations.

The action parameter of `crudUrl` is used to get the corresponding CRUD action.

The following code example describes the above behavior.

```
`ts
import { Grid, Edit, Toolbar } from '@syncfusion/ej2-grids';
import { DataManager, UrlAdaptor } from '@syncfusion/ej2-data';
Grid.Inject(Edit, Toolbar);
```

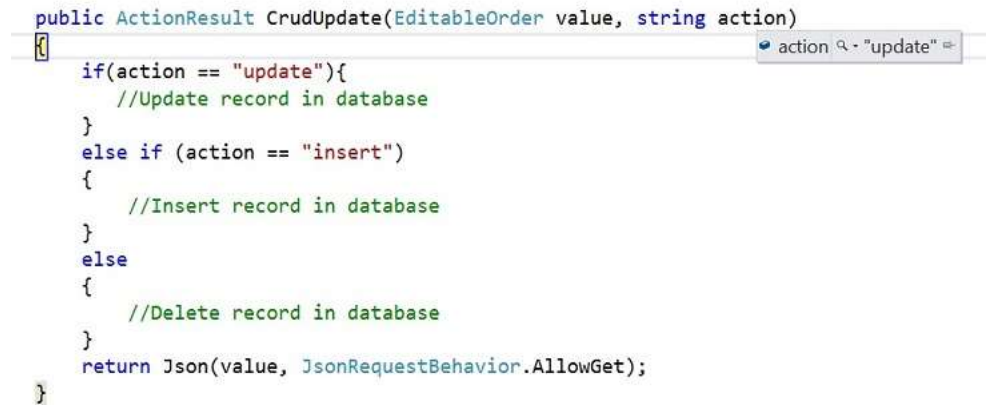
```

let data: DataManager = new DataManager({
url: "Home/DataSource",
crudUrl: "Home/CrudUpdate",
adaptor: new UrlAdaptor
});
let grid: Grid = new Grid({
dataSource: data,
toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
editSettings: { allowEditing: true, allowAdding: true, allowDeleting: true, mode: 'Normal' },
columns: [
{ field: 'OrderID', headerText: 'Order ID', textAlign: 'Right', width: 100, isPrimaryKey: true,
validationRules: { required: true } },
{ field: 'CustomerID', headerText: 'Customer ID', width: 120, validationRules: { required: true, minLength:
3 }, defaultValue: 'HANAR' },
{ field: 'Freight', headerText: 'Freight', textAlign: 'Right', editType: 'numericedit', width: 120, format: 'C2'
},
{ field: 'ShipCountry', headerText: 'Ship Country', editType: 'dropdownedit', width: 150 }
],
height: 265
});
grid.appendTo('#Grid');
`ts
public ActionResult CrudUpdate(EditableOrder value, string action)
{
if(action == "update"){
//Update record in database
}
else if (action == "insert")
{
//Insert record in database
}
else
{

```

```
//Delete record in database
}
}
,
```

Please refer to the following screenshot to know about the action parameter.



```
public ActionResult CrudUpdate(EditableOrder value, string action)
{
    if(action == "update"){
        //Update record in database
    }
    else if (action == "insert")
    {
        //Insert record in database
    }
    else
    {
        //Delete record in database
    }
    return Json(value, JsonRequestBehavior.AllowGet);
}
```

If you specify `insertUrl` along with `crudUrl`, then while adding `insertUrl` only will be invoked.

Batch URL

The `batchUrl` property supports only for batch editing mode. You can specify the controller action mapping URL to perform batch operation on the server-side.

The following code example describes the above behavior.

```
`ts
import { Grid, Edit, Toolbar } from '@syncfusion/ej2-grids';
import { DataManager, UrlAdaptor } from '@syncfusion/ej2-data';
Grid.Inject(Edit, Toolbar);
let data: DataManager = new DataManager({
    url: "Home/DataSource",
    batchUrl: "Home/BatchUpdate",
    adaptor: new UrlAdaptor
});
let grid: Grid = new Grid({
    dataSource: data,
    toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
    editSettings: { allowEditing: true, allowAdding: true, allowDeleting: true, mode: 'Batch' },
    columns: [
```

```

{ field: 'OrderID', headerText: 'Order ID', textAlign: 'Right', width: 100, isPrimaryKey: true,
validationRules: { required: true } },

{ field: 'CustomerID', headerText: 'Customer ID', width: 120, validationRules: { required: true, minLength:
3 }, defaultValue: 'HANAR' },

{ field: 'Freight', headerText: 'Freight', textAlign: 'Right', editType: 'numericedit', width: 120, format: 'C2'
},

{ field: 'ShipCountry', headerText: 'Ship Country', editType: 'dropdownedit', width: 150 }
],
height: 265
});

grid.appendTo('#Grid');
`
`ts

public ActionResult BatchUpdate(string action, List<EditableOrder> added, List<EditableOrder> changed,
List<EditableOrder> deleted, int? key)
{
//Save the batch changes in database
}

`

public ActionResult BatchUpdate(string action, List<EditableOrder> added, List<EditableOrder> changed, List<EditableOrder> deleted, int? key)
{
//Save the batch changes in database
return Json(changed, JsonRequestBehavior.AllowGet);
}

```

changed: Count = 2

CustomerID	"HANAR-Updated"
EmployeeID	4
Freight	65.83
OrderDate	(7/8/1996 5:30:00 AM)
OrderID	10250
ShipAddress	"Rua do Paço, 67"
ShipCity	"Rio de Janeiro"
ShipCountry	"Brazil"
ShipName	"Hanari Carnes"
ShipPostalCode	"05454-876"
ShipRegion	"RJ"
Verified	true

Sorting in ##Platform_Name## Grid control

Sorting enables you to sort data in the **Ascending** or **Descending** order. To sort a column, click the column header.

To sort multiple columns, press and hold the CTRL key and click the column header. You can clear sorting of any one of the multi-sorted columns by pressing and holding the SHIFT key and clicking the specific column header.

To enable batch sorting in the Grid, set the [allowSorting](#) to true. Sorting options can be configured through the [sortSettings](#).

To sort, inject the [Sort](#) module in the grid.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Sort);
var grid = new ej.grids.Grid({

```



```

        dataSource: data,
        allowSorting: true,
        columns: [
            { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
            { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
            { field: 'ShipCity', headerText: 'Ship City', width: 150 },
            { field: 'ShipName', headerText: 'Ship Name', width: 150 }
        ],
        height: 315
    });
    grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <style>
        .e-row[aria-selected="true"] .e-customizedExpandcell {
            background-color: #e0e0e0;
        }
        .e-grid.e-gridhover tr[role='row']:hover {
            background-color: #eee;
        }
        .e-expand::before {

```

```

        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

* Grid columns are sorted in the **Ascending** order. If you click the already sorted column, the sort direction toggles.

* You can apply and clear sorting by invoking [sortColumn](#) and [clearSorting](#) methods.

* To disable sorting for a particular column, set the [columns.allowSorting](#) to false.

Initial sort

To sort at initial rendering, set the [field](#) and [direction](#) in the **sortSettings.columns**.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Sort);
var grid = new ej.grids.Grid({
    dataSource: data,
    allowSorting: true,
    sortSettings: { columns: [{ field: 'OrderID', direction: 'Ascending' },
    { field: 'ShipCity', direction: 'Descending' }] },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'ShipCity', headerText: 'Ship City', width: 150 },
        { field: 'ShipName', headerText: 'Ship Name', width: 150 }
    ],
    height: 315
});

```

```
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
      background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
      background-color: #eee;
    }
    .e-expand::before {
      content: '\e5b8';
    }
    .empImage {
      margin: 6px 16px;
      float: left;
      width: 50px;
      height: 50px;
    }
  </style>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
```

```

<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Multi-column sorting

You can sort more than one column in a Grid. To sort multiple columns, press and hold the **CTRL** key and click the column header. The sorting order will be displayed in the header while performing multi-column sorting.

To clear sorting for a particular column, press the "Shift + mouse left click".

The [allowSorting](#) must be true while enabling multi-column sort.

Set [allowMultiSorting](#) property as **false** to disable multi-column sorting.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Sort);
var grid = new ej.grids.Grid({
    dataSource: data,
    allowSorting: true,
    allowMultiSorting: true,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'ShipCity', headerText: 'Ship City', width: 150 },
        { field: 'ShipName', headerText: 'Ship Name', width: 150 }
    ],
    height: 315
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {

```

```

    ele.style.visibility = "visible";
  }
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Sort order

By default, the sorting order will be as **ascending -> descending -> none**.

When first click a column header it sorts the column in ascending. Again click the same column header, it will sort the column in descending. A repetitive third click on the same column header will clear the sorting.

Sort foreign key column based on Text

For local data in Grid, sorting will be performed based on the [foreignKeyValue](#).

For remote data in Grid, sorting will be performed based on the [foreignKeyField](#), we need to handle the sorting operation at the server side.

`ts

```

import { Grid, ForeignKey, Sort } from '@syncfusion/ej2-grids';
import { DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';

Grid.Inject(ForeignKey, Sort);

let dataSource: DataManager = new DataManager({
  json: '/OData/Items',
  adaptor: new ODataV4Adaptor
});

let employeeData: DataManager = new DataManager({
  url: '/OData/Brands',
  adaptor: new ODataV4Adaptor
});

let grid: Grid = new Grid(
{
  dataSource: data,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right', width: 100 },
    {
      field: 'EmployeeID', headerText: 'Employee Name', width: 150, foreignKeyValue: 'FirstName',
      dataSource: employeeData
    },
    { field: 'Freight', headerText: 'Freight', width: 100, textAlign: 'Right'},

```

```
{ field: 'ShipName', headerText: 'Ship Name', width: 180 }
],
height: 315
});
grid.appendTo('#Grid');
`
```

The following code example describes the handling of sorting operation at the server side.

```
`
public class ItemsController : ODataController
{
    [EnableQuery]
    public IQueryable<Item> Get()
    {
        List<Item> GridData =
        JsonConvert.DeserializeObject<Item[]>(Properties.Resources.ItemsJson).AsQueryable().ToList();
        List<Brand> empData =
        JsonConvert.DeserializeObject<Brand[]>(Properties.Resources.BrandsJson).AsQueryable().ToList();
        var queryString = HttpContext.Current.Request.QueryString;
        var allUrlKeyValues = ControllerContext.Request.GetQueryNameValuePairs();
        string key = allUrlKeyValues.LastOrDefault(x => x.Key == "$orderby").Value;
        if (key != null)
        {
            if (key == "EmployeeID") {
                GridData = SortFor(key); //Only for foreignKey Column ascending
            }
            else if(key == "EmployeeID desc") {
                GridData = SortFor(key); //Only for foreignKey Column descending
            }
        }
        var count = GridData.Count();
        var data = GridData.AsQueryable();
        return data;
    }
    public List<Item> SortFor(String Sorted)
`
```

```

{
List<Item> GridData =
JsonConvert.DeserializeObject<Item[]>(Properties.Resources.ItemsJson).AsQueryable().ToList();

List<Brand> empData =
JsonConvert.DeserializeObject<Brand[]>(Properties.Resources.BrandsJson).AsQueryable().ToList();

if (Sorted == "EmployeeID") //check whether ascending or descending
empData = empData.OrderBy(e => e.FirstName).ToList();
else if(Sorted == "EmployeeID desc")
empData = empData.OrderByDescending(e => e.FirstName).ToList();

List<Item> or = new List<Item>();
for (int i = 0; i < empData.Count(); i++) {
//Select the Field matching records

IEnumerable<Item> list = GridData.Where(pred => pred.EmployeeID ==
empData[i].EmployeeID).ToList();
or.AddRange(list);
}
return or;
}
}
,

```

Sorting events

During the sort action, the grid component triggers two events. The [actionBegin](#) event triggers before the sort action starts, and the [actionComplete](#) event triggers after the sort action is completed. Using these events you can perform the needed actions.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Sort);
var grid = new ej.grids.Grid({
  dataSource: data,
  allowSorting: true,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
    { field: 'ShipCity', headerText: 'Ship City', width: 150 },
    { field: 'ShipName', headerText: 'Ship Name', width: 150 }
  ],
  height: 315,
  actionBegin: actionHandler,
  actionComplete: actionHandler
});
grid.appendTo('#Grid');
function actionHandler(args) {

```



```
    alert(args.requestType + ' ' + args.type); //custom Action
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Grid"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
```

```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

The [args.requestType](#) is the current action name. For example, in sorting the [args.requestType](#) value is 'sorting'.

Custom sort comparer

You can customize the default sort action for a column by defining the [column.sortComparer](#) property. The sort comparer function has the same functionality like [Array.sort](#) sort comparer.

In the following example, custom sort comparer function was defined in the **Customer ID** column.

INDEX.JS

```
function sortComparer(reference, comparer) {
    if (reference == 32.38) {
        return -1;
    }
    return reference - comparer;
};
ej.grids.Grid.Inject(ej.grids.Sort);
var grid = new ej.grids.Grid({
    dataSource: data,
    allowSorting: true,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign:
'Right', width: 120 },
        { field: 'CustomerID', width: 140, headerText: 'Customer ID'
},
        { field: 'Freight', headerText: 'Freight', textAlign:
'Right', width: 120, format: 'C2', sortComparer: sortComparer },
        { field: 'OrderDate', headerText: 'Order Date', textAlign:
'Right', width: 140, format: 'yMd' }
    ],
    height: 315
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="dropDown"></div>
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The sort comparer function will work only for the local data.

Touch interaction

When you tap the grid header on touchscreen devices, the selected column header is sorted. A popup

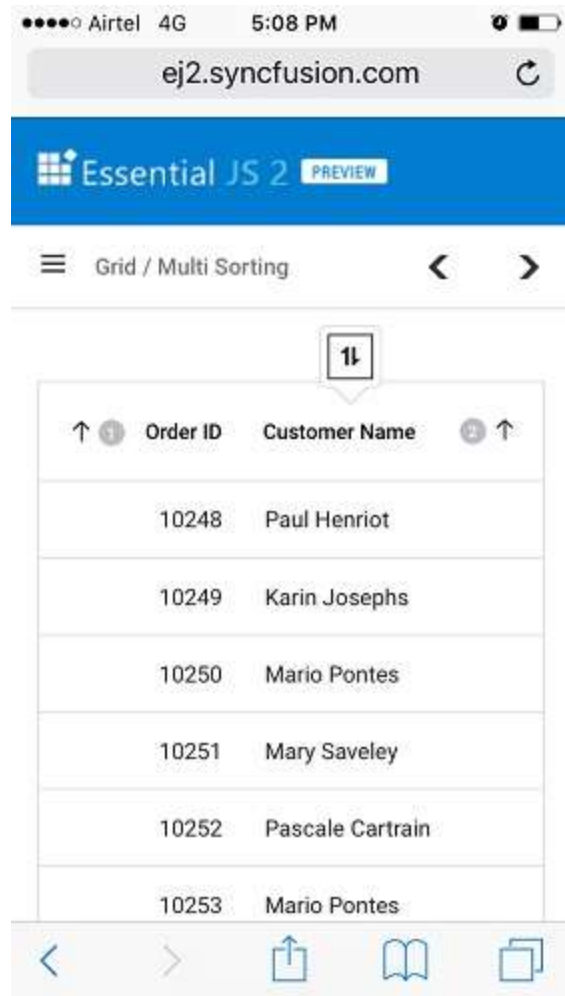


is displayed for multi-column sorting. To sort multiple columns, tap the popup , and then tap the desired grid headers.



The [allowMultiSorting](#) and [allowSorting](#) should be `true` then only the popup will be shown.

The following screenshot shows grid touch sorting.



Grouping

Grouping in ##Platform_Name## Grid control

The Grid has options to group records by dragging and dropping the column header to the group drop area. When grouping is applied, grid records are organized into a hierarchical structure to facilitate easier expansion and collapse of records.

To enable grouping in the grid, set the [allowGrouping](#) as true. Grouping options can be configured through the [groupSettings](#). To group, inject the [Group](#) module in the grid.

INDEX.JS

```
ej.grids.Grid.Inject(ej.grids.Sort, ej.grids.Group);
var grid = new ej.grids.Grid({
  dataSource: data,
  allowGrouping: true,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
    { field: 'ShipCity', headerText: 'Ship City', width: 150 },
    { field: 'ShipName', headerText: 'Ship Name', width: 150 }
  ],
  height: 267
});
```

```
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
      background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
      background-color: #eee;
    }
    .e-expand::before {
      content: '\e5b8';
    }
    .empImage {
      margin: 6px 16px;
      float: left;
      width: 50px;
      height: 50px;
    }
  </style>
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

* You can group and ungroup columns by using the [groupColumn](#) and [ungroupColumn](#) methods.

* To disable grouping for a particular column, set the [columns.allowGrouping](#) to false.

Initial group

To apply group at initial rendering, set the column field name in the `groupSettings.columns`.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Sort, ej.grids.Group);
var grid = new ej.grids.Grid({
    dataSource: data,
    allowGrouping: true,
    groupSettings: { columns: ['CustomerID', 'ShipCity'] },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'ShipCity', headerText: 'Ship City', width: 150 },
        { field: 'ShipName', headerText: 'Ship Name', width: 150 }
    ],
    height: 267
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}

```

```

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Hide drop area

To avoid ungrouping or further grouping of a column after initial column grouping, define the [groupSettings.showDropArea](#) as false.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Sort, ej.grids.Group);
var grid = new ej.grids.Grid({
  dataSource: data,
  allowGrouping: true,
  groupSettings: { showDropArea: false, columns: ['CustomerID',
'ShipCity'] },
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
    { field: 'ShipCity', headerText: 'Ship City', width: 150 },
    { field: 'ShipName', headerText: 'Ship Name', width: 150 }
  ],
  height: 315
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Group with paging

On grouping columns with paging feature, the aggregated information and total items are displayed based on the current page. The grid does not consider aggregated information and total items from other pages. To get additional details (aggregated information and total items) from other pages, set the [groupSettings.disablePageWiseAggregates](#) to false.

If remote data is bound to grid dataSource, two requests will be sent when performing grouping action; one for getting the grouped data and another for getting aggregate details and total items count.

Group by format

By default, a column will be grouped by the data or value present for the particular row. To group the numeric or datetime column based on the mentioned format, you have to enable the [enableGroupByFormat](#) property of the corresponding

grid columns.

INDEX.JS

```
ej.grids.Grid.Inject(ej.grids.Sort, ej.grids.Group);
var grid = new ej.grids.Grid({
  dataSource: data,
  allowGrouping: true,
  groupSettings: { showDropArea: false, columns: ['OrderDate', 'Freight']
},
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
    { field: 'OrderDate', headerText: 'Order Date', format: 'yM',
enableGroupByFormat: true, width: 150 },
    { field: 'Freight', headerText: 'Freight', format: 'C2',
enableGroupByFormat: true, width: 150 }
  ],
  height: 315
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Grouping events

During the group action, the grid component triggers two events. The [actionBegin](#) event triggers before the group action starts and the [actionComplete](#) event triggers after the group action is completed. Using these events you can perform any action.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Group, ej.grids.Sort);
var grid = new ej.grids.Grid({
    dataSource: data,
    allowGrouping: true,

```

```

        columns: [
            { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
            { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
            { field: 'ShipCity', headerText: 'Ship City', width: 150 },
            { field: 'ShipName', headerText: 'Ship Name', width: 150 }
        ],
        height: 260,
        actionBegin: actionHandler,
        actionComplete: actionHandler
    });
    grid.appendTo('#Grid');
    function actionHandler(args) {
        alert(args.requestType + ' ' + args.type); //Custom Action
    }

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>

```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The [args.requestType](#) is based on the current action name. For example, when grouping, the [args.requestType](#) value will be 'grouping'.

Collapse by external button

You can collapse the selected group from an external button by invoking the [expandCollapseRows](#) method.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Group, ej.grids.Filter, ej.grids.Sort);
var grid = new ej.grids.Grid({
    dataSource: data,
    allowGrouping: true,
    groupSettings: { columns: ['ShipCity'] },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'ShipCity', headerText: 'Ship City', width: 150 },
        { field: 'ShipName', headerText: 'Ship Name', width: 150 }
    ],
    height: 240
});
grid.appendTo('#Grid');
var collapse = new ej.buttons.Button({ cssClass: 'e-flat' }, '#collapse');
document.getElementById('collapse').addEventListener('click', function() {
    if (grid.getSelectedRowIndex().length) {
        var currentTr = grid.getRows()[grid.getSelectedRowIndex()[0]];
        while (currentTr.classList && currentTr.classList.length) {
            currentTr = currentTr.previousSibling;
        }
        var collapseElement = currentTr.querySelector('.e-
recordplusexpand');
        grid.groupModule.expandCollapseRows(collapseElement); //Pass the
collapse row element.
    }
});

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <button id="collapse">Collapse</button>
    <div id="Grid"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Sort grouped columns in descending order during initial grouping

By default, grouped columns are sorted in ascending order. To sort grouped columns in descending order during initial grouping, you can set the [field](#) and [direction](#) in the `sortSettings.columns` property.

The `CustomerID` column will be sorted in descending order when the grid is initially grouped, as shown in the following example.

INDEX.JS

```
var grid = new ej.grids.Grid({
  dataSource: data,
  allowGrouping: true,
  allowSorting: true,
  sortSettings: { columns: [{ field: 'CustomerID', direction: 'Descending'
}] },
  groupSettings: { columns: ['CustomerID'] },
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
    { field: 'ShipCity', headerText: 'Ship City', width: 150 },
    { field: 'ShipName', headerText: 'Ship Name', width: 150 }
  ],
  height: 267
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

[See Also](#)

Caption template in ##Platform_Name## Grid control

You can customize the group caption by using the [groupSettings.captionTemplate](#) property.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Group);
var grid = new ej.grids.Grid({
    dataSource: employeeData,
    allowGrouping: true,
    groupSettings: { captionTemplate: '#captiontemplate', columns:
['EmployeeID'] },
    columns: [

```



```

        { field: 'EmployeeID', headerText: 'Employee ID' },
        { field: 'CustomerID', headerText: 'Customer ID' },
        { field: 'FirstName', headerText: 'Name', width: 120 },
        { field: 'Title', headerText: 'Title', width: 170 }
    ],
    height: 315
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <script id="captiontemplate" type="text/x-template">
            ${field} - ${key}

```

```

        </script>
        <div id="Grid"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Adding custom text in group caption

You can customize the group caption text by using the [groupSettings.captionTemplate](#) property.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Group);
var grid = new ej.grids.Grid({
    dataSource: employeeData,
    allowGrouping: true,
    groupSettings: { captionTemplate: '#captiontemplate', columns:
    ['EmployeeID'] },
    columns: [
        { field: 'EmployeeID', headerText: 'Employee ID' },
        { field: 'CustomerID', headerText: 'Customer ID' },
        { field: 'FirstName', headerText: 'Name', width: 120 },
        { field: 'Title', headerText: 'Title', width: 170 }
    ],
    height: 315
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <script id="captiontemplate" type="text/x-template">
            <div>${template(data)}</div>
        </script>
        <div id="Grid"></div>
    </div>
    <script type="text/javascript">
        function template(args) {
            //you can add add customtext here.
            return args.key + " :" + args.count + " custom text" ;
        }
    </script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Render custom component in group caption

We can render custom components in the group caption by using the [groupSettings.captionTemplate](#) property.

In the below demo, we have rendered the EJ2 [ButtonComponent](#) in the group caption.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Group, ej.grids.Page);
var grid = new ej.grids.Grid({
    dataSource: data,
    allowPaging: true,
    dataBound: () => {

```

```

        let groupCations =
document.getElementsByClassName('groupbutton');
        for(var i=0;i<groupCations.length;i++) {
            var button = new ej.buttons.Button({
                isPrimary: true
            });
            button.appendTo(groupCations[i]);
        }
    },
    height: 315,
    allowGrouping: true,
    groupSettings: {columns: ["ShipCountry"], captionTemplate:
"#captiontemplate"},
    columns: [
        { field: 'OrderID', headerText: 'Order ID', width: 120,
textAlign: 'Right' },
        { field: 'CustomerID', headerText: 'Customer Name', width: 150
},
        { field: 'OrderDate', headerText: 'Order Date', width: 130,
format: 'yMd', textAlign: 'Right' },
        { field: 'Freight', width: 120, format: 'C2', textAlign: 'Right'
},
        { field: 'ShipCountry', headerText: 'Ship Country', width: 150 }
    ]
    });
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-grids/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
    <script id="captiontemplate" type="text/x-template">
        ${field} - ${key} - <button class='groupbutton'>${count}
items</button>
    </script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Reordering on grouped columns in ##Platform_Name## Grid control

Grid provides an option to interchange the position of the Grouped Columns by dragging and dropping the grouped headercells in the droparea. So, any new column entering into the droparea can also be dropped in any position.

To enable this feature, you have to set the [groupSettings.allowReordering](#) property as **true**.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Group, ej.grids.Filter, ej.grids.Sort);
var grid = new ej.grids.Grid({
    dataSource: data,
    allowGrouping: true,
    groupSettings: { columns: ['ShipCity'], allowReordering: true },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'ShipCity', headerText: 'Ship City', width: 150 },
        { field: 'ShipName', headerText: 'Ship Name', width: 150 }
    ],
    height: 240
});
grid.appendTo('#Grid');

```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Grid"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Lazy load grouping in ##Platform_Name## Grid control

The lazy load grouping allows you to load grouped records to the Grid through the on-demand concept. So, you can use this feature to load a huge amount of grouped data to the Grid without any performance degradation.

When you enable this feature, the Grid will render only the initial level caption rows in the collapsed state at grouping. The child rows of each caption will be fetched from the server and render in the Grid when you expand the caption row. The fetching child records count will be implicitly determined by the content area occupying rows count. So, if the child records exceed the count, then the Grid will request the server again to fetch the next block of child records on scrolling.

The caption row expand/collapse state will be persisted on paging and Grid pages count will be determined based on the caption rows count instead of the child rows.

To enable this feature, you have to set the [groupSettings.enableLazyLoading](#) property as **true**.

INDEX.JS

```
ej.grids.Grid.Inject(ej.grids.Page, ej.grids.Group, ej.grids.LazyLoadGroup);
var grid = new ej.grids.Grid({
  dataSource: data,
  allowPaging: true,
  allowGrouping: true,
  groupSettings: { enableLazyLoading: true, columns: ['ProductName',
'CustomerName'] },
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
    { field: 'ProductName', headerText: 'Product Name', width: 160 },
    { field: 'ProductID', headerText: 'Product ID', textAlign: 'Right',
width: 120 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
    { field: 'CustomerName', headerText: 'Customer Name', width: 160 }
  ],
  height: 220
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Handling the lazy load grouping at server-side

You can use the [UrlAdaptor](#) of [DataManager](#) when binding the remote data. Along with the default server request, this feature will additionally send the below details to handle the lazy load grouping. In the server end, these details are bound with the [ISLazyLoad](#) and [OnDemandGroupInfo](#) parameters in the [DataManagerRequest](#) model. Please refer to the below table and screenshots.

Property Name | Description

[isLazyLoad](#) | To differentiate the default grouping and lazy load grouping

[onDemandGroupInfo](#) | Have the details of expanded caption row grouping [level](#), [skip](#), [take](#) and [filter](#) query of the child records


```

.....if (dm.IsLazyLoad)
.....{
.....    dm.IsLazyLoad = true;
.....    groupedData = operation.PerformGrouping<Customers>(DataSource, dm); // Lazy load grouping
.....    if (dm.OnDemandGroupInfo != null && dm.Group.Count() == dm.OnDemandGroupInfo.Level)
.....    {
.....        count = groupedData.Cast<Customers>().Count();
.....    }
.....    else
.....    {
.....        count = groupedData.Cast<Group>().Count();
.....    }
.....    groupedData = operation.PerformSkip(groupedData, dm.OnDemandGroupInfo == null ? dm.Skip : dm.OnDemandGroupInfo.Skip);
.....    groupedData = operation.PerformTake(groupedData, dm.OnDemandGroupInfo == null ? dm.Take : dm.OnDemandGroupInfo.Take);
.....    return dm.RequiresCounts ? Json(new { result = groupedData == null ? DataSource : groupedData, count = count }) : Json(DataSource);
.....}

.....if (dm.IsLazyLoad)
.....{
.....    groupedData = operation.PerformGrouping<Customers>(DataSource, dm); // Lazy load grouping
.....    if (dm.OnDemandGroupInfo != null && dm.Group.Count() == dm.OnDemandGroupInfo.Level)
.....    {
.....        dm.OnDemandGroupInfo = (Syncfusion.EJ2.Base.OnDemandGroupInfo)
.....        {
.....            Level: 1,
.....            Skip: 0,
.....            Take: 33,
.....            Where: Count = 1
.....        };
.....        count = groupedData.Cast<Customers>().Count();
.....    }
.....    else
.....    {
.....        count = groupedData.Cast<Group>().Count();
.....    }
.....    groupedData = operation.PerformSkip(groupedData, dm.OnDemandGroupInfo == null ? dm.Skip : dm.OnDemandGroupInfo.Skip);
.....    groupedData = operation.PerformTake(groupedData, dm.OnDemandGroupInfo == null ? dm.Take : dm.OnDemandGroupInfo.Take);
.....    return dm.RequiresCounts ? Json(new { result = groupedData == null ? DataSource : groupedData, count = count }) : Json(DataSource);
.....}

```

The following code example describes the lazy load grouping handled at the server-side with other grid actions.

```
`ts
```

```

public IActionResult UrlDatasource([FromBody] DataManagerRequest dm)
{
    IEnumerable groupedData = null;
    IEnumerable<Customers> DataSource = customers;
    DataOperations operation = new DataOperations();
    if (dm.Search != null && dm.Search.Count > 0)
    {
        DataSource = operation.PerformSearching(DataSource, dm.Search); //Search
    }
    if (dm.Sorted != null && dm.Sorted.Count > 0) //Sorting
    {
        DataSource = operation.PerformSorting(DataSource, dm.Sorted);
    }
    if (dm.Where != null && dm.Where.Count > 0) //Filtering
    {
        DataSource = operation.PerformFiltering(DataSource, dm.Where, dm.Where[0].Operator);
    }
    int count = DataSource.Cast<Customers>().Count();
    if (dm.IsLazyLoad == false && dm.Skip != 0)
    {

```

```

DataSource = operation.PerformSkip(DataSource, dm.Skip); // Paging
}
if (dm.IsLazyLoad == false && dm.Take != 0)
{
    DataSource = operation.PerformTake(DataSource, dm.Take);
}
if (dm.IsLazyLoad)
{
    groupedData = operation.PerformGrouping<Customers>(DataSource, dm); // Lazy load grouping
    if (dm.OnDemandGroupInfo != null && dm.Group.Count() == dm.OnDemandGroupInfo.Level)
    {
        count = groupedData.Cast<Customers>().Count();
    }
    else
    {
        count = groupedData.Cast<Group>().Count();
    }
    groupedData = operation.PerformSkip(groupedData, dm.OnDemandGroupInfo == null ? dm.Skip :
    dm.OnDemandGroupInfo.Skip);
    groupedData = operation.PerformTake(groupedData, dm.OnDemandGroupInfo == null ? dm.Take :
    dm.OnDemandGroupInfo.Take);
}
return dm.RequiresCounts ? Json(new { result = groupedData == null ? DataSource : groupedData, count
= count }) : Json(DataSource);
}
`

```

Lazy load grouping with infinite scrolling

Infinite scrolling loads a huge amount of data without degrading the Grid's performance. By default, infinite scrolling is enabled only for the expanded grouped rows when lazy loading is enabled. Now, the Grid has an option to allow infinite scrolling for group caption rows. This is achieved by setting the [enableInfiniteScrolling](#) property as true when lazy loading is enabled in the grouped records.

This is demonstrated in the following sample:

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Page, ej.grids.Group, ej.grids.LazyLoadGroup,
ej.grids.InfiniteScroll);
var grid = new ej.grids.Grid({
    dataSource: data,

```

```

enableInfiniteScrolling: true,
allowGrouping: true,
groupSettings: { enableLazyLoading: true, columns: ['ProductName',
'CustomerName'] },
columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
    { field: 'ProductName', headerText: 'Product Name', width: 160 },
    { field: 'ProductID', headerText: 'Product ID', textAlign: 'Right',
width: 120 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
    { field: 'CustomerName', headerText: 'Customer Name', width: 160 }
],
height: 220
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>

```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The Grid also supports the lazy load grouping with [virtualization](#)(virtual scrolling).

Limitations for lazy load grouping

- Due to the element height limitation in browsers, the maximum number of records loaded by the grid is limited due to the browser capability.
- DataManager's `UrlAdaptor` and `JsonAdaptor` only have the built-in lazy load grouping support.
- Lazy load grouping is not compatible with batch editing, row template etc.
- Programmatic selection is not supported.

Filtering

Filtering in ##Platform_Name## Grid control

Filtering allows you to view particular records based on filter criteria. To enable filtering in the Grid, set the [allowFiltering](#) to true. Filtering options can be configured through [filterSettings](#).

To use filter, inject the [Filter](#) module in the grid.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Filter);
var grid = new ej.grids.Grid({
    dataSource: data,
    allowFiltering: true,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
        { field: 'ShipCity', headerText: 'Ship City', width: 100 },
        { field: 'ShipName', headerText: 'Ship Name', width: 100 }
    ],
    height: 273
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>

```

```

<title>EJ2 Grid</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```

```

<div id="container">
  <div id="Grid"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

* You can apply and clear filtering by using [filterByColumn](#) and [clearFiltering](#) methods.

* To disable filtering for a particular column, set [columns.allowFiltering](#) to false.

Initial filter

To apply the filter at initial rendering, set the filter [predicate](#) object in [filterSettings.columns](#).

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Filter);
var grid = new ej.grids.Grid({
  dataSource: data,
  allowFiltering: true,
  filterSettings: {
    columns: [{ field: 'ShipCity', matchCase: false, operator:
'startswith', predicate: 'and', value: 'reims' },
    { field: 'ShipName', matchCase: false, operator: 'startswith',
predicate: 'and', value: 'Vins et alcools Chevalier' }]
  },
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
    { field: 'ShipCity', headerText: 'Ship City', width: 100 },
    { field: 'ShipName', headerText: 'Ship Name', width: 100 }
  ],
  height: 273
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>

```

```
</body></html>
```

Filter operators

The filter operator for a column can be defined in the [filterSettings.columns.operator](#) property.

The available operators and its supported data types are:

Operator | Description | Supported Types

a*b | Everything that starts with "a" and ends with "b".

a* | Everything that starts with "a".

*b | Everything that ends with "b".

a | Everything that has an "a" in it.

ab* | Everything that has an "a" in it, followed by anything, followed by a "b", followed by anything.

Search			Columns	
Order ID		Shipped Date		Ship Country
10250		7/12/1996		Brazil
10251		7/15/1996		France
10252		7/11/1996		Belgium
10253		7/16/1996		Brazil
10254		7/23/1996		Switzerland
10255		7/15/1996		Switzerland
10256		7/17/1996		Brazil
10257		7/22/1996		Venezuela
10258		7/23/1996		Austria

LIKE filtering

The **LIKE** filter can process single search patterns using the "%" symbol, retrieving values matching the specified patterns. The following Grid features support LIKE filtering on string-type columns:

- Filter Menu
- Filter Bar with the [filterSettings.showFilterBarOperator](#) property enabled on the Grid [filterSettings](#).

- Custom Filter of Excel filter type.

For example:

Operator | Description

%ab% | Returns all the value that are contains "ab" character.

ab% | Returns all the value that are ends with "ab" character.

%ab | Returns all the value that are starts with "ab" character.

Order ID	Shipped Date	Ship Country
10250	7/12/1996	Brazil
10251	7/15/1996	France
10252	7/11/1996	Belgium
10253	7/16/1996	Brazil
10254	7/23/1996	Switzerland
10255	7/15/1996	Switzerland
10256	7/17/1996	Brazil
10257	7/22/1996	Venezuela
10258	7/23/1996	Austria

By default, the [filterSettings.columns.operator](#) value is **equal**.

Diacritics filter

By default, grid ignores diacritic characters while filtering. To include diacritic characters, set the [filterSettings.ignoreAccent](#) as **true**.

In the following sample, type **aero** in **Name** column to filter diacritic characters.

INDEX.JS

```
ej.grids.Grid.Inject(ej.grids.Filter);
var grid = new ej.grids.Grid({
  dataSource: data,
  allowFiltering: true,
  filterSettings: { ignoreAccent: true },
```

```

        columns: [
            { field: 'EmployeeID', headerText: 'Employee ID', textAlign:
'Right', width: 140 },
            { field: 'Name', headerText: 'Name', width: 170 },
            { field: 'ShipName', headerText: 'Ship Name', width: 170 },
            { field: 'CustomerID', headerText: 'Supplier Name', width: 140 }
        ],
        allowPaging: true,
    });
    grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">

```

```

        <div id="Grid"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Customizing Filter Dialog by using an additional parameter](#)
- [How to implement 'not contains' operator for Grid filtering](#)
- [How to filter custom date ranges in Grid column using date range picker](#)
- [How to filter multiple records using the filter bar template](#)
- [How to change the data source for checkbox filter popup in Grid?](#)
- [How to perform advanced filtering in grid using custom queries](#)

Filter bar in ##Platform_Name## Grid control

By setting the [allowFiltering](#) to true, the filter bar row will render next to the header, which allows you to filter data. You can filter the records with different expressions depending upon the column type.

Filter bar expressions:

You can enter the following filter expressions (operators) manually in the filter bar.

Expression	Example	Description	Column Type
------------	---------	-------------	-------------

=	=value	equal	Number
---	--------	-------	--------

!=	!=value	notequal	Number
----	---------	----------	--------

>	>value	greaterthan	Number
---	--------	-------------	--------

<	<value	lessthan	Number
---	--------	----------	--------

=	>=value	greaterthanorequal	Number
---	---------	--------------------	--------

<=	<=value	lessthanorequal	Number
----	---------	-----------------	--------

/	/value	startswith	String
---	--------	------------	--------

%	%value	endswith	String
---	--------	----------	--------

N/A	N/A	Equal	operator will always be used for date filter. Date
-----	-----	-------	---

N/A	N/A	Equal	operator will always be used for Boolean filter. Boolean
-----	-----	-------	---

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Filter);
var grid = new ej.grids.Grid({
    dataSource: data,
    allowFiltering: true,
    columns: [

```

```

        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
        { field: 'ShipCity', headerText: 'Ship City', width: 100 },
        { field: 'ShipName', headerText: 'Ship Name', width: 100 }
    ],
    height: 273
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <style>
        .e-row[aria-selected="true"] .e-customizedExpandcell {
            background-color: #e0e0e0;
        }
        .e-grid.e-gridhover tr[role='row']:hover {
            background-color: #eee;
        }
        .e-expand::before {
            content: '\e5b8';
        }
        .empImage {

```

```

        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Filter bar template with custom component

The [filterBarTemplate](#) is used to add custom components to a particular column, and does the following functions:

- **create**: Creates custom components.
- **write**: Wires events for custom components.

In the following sample, the dropdown is used as a custom component in the EmployeeID column.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Filter);
var gridObj = new ej.grids.Grid({
    dataSource: data,
    allowFiltering: true,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100 },
        {
            field: 'EmployeeID', filterBarTemplate: {
                create: function(args){
                    var dd = document.createElement('input');
                    dd.id = 'EmployeeID';
                    return dd;
                },
                write: function(args){
                    var DropDownListObj = new ej.dropdowns.DropDownList({
                        dataSource: ['All', '1', '3', '4', '5', '6', '8', '9'],
                        fields: { text: 'EmployeeID', value: 'EmployeeID' },

```

```

                placeholder: 'Select a value',
                popupHeight: '200px',
                change: function(e) {
                    var gridObj =
(document.getElementsByClassName('e-grid')[0]).ej2_instances[0];
                    if (e.value == 'All') {

gridObj.removeFilteredColsByField('EmployeeID');
                        } else {

gridObj.filterByColumn('EmployeeID', 'equal', parseInt(e.value));
                        }
                    }
                })
                DropDownListObj.appendTo('#EmployeeID');
            },
            textAlign: 'Right', width: 100
        },
        { field: 'ShipCity', headerText: 'Ship City', width: 100 },
        { field: 'ShipName', headerText: 'Ship Name', width: 100 }
    ]
});
gridObj.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Change default filterbar operator

You can change the filter operator as per the requirement by defining the **filter** property in Grid columns.

In this below demo, we have applied the filter operator **contains** for CustomerID column.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Page, ej.grids.Filter);
var grid = new ej.grids.Grid({
    dataSource: data,
    allowPaging: true,
    allowFiltering: true,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
        { field: 'CustomerID', width: 140, filter: { operator: "contains" },
headerText: 'Customer ID', type: 'string' },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C2' },
        { field: 'OrderDate', headerText: 'Order Date', textAlign: 'Right',
width: 140, format: 'yMd' }
    ],
    height: 220
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
<title>EJ2 Grid</title>

```

```

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See also

- [How to perform filter by using Wildcard and LIKE operator filter](#)

Filter menu in ##Platform_Name## Grid control

You can enable filter menu by setting the [filterSettings.type](#) as **Menu**. The filter menu UI will be rendered based on its column type, which allows you to filter data. You can filter the records with different operators.

INDEX.JS

```
ej.grids.Grid.Inject(ej.grids.Filter);
var grid = new ej.grids.Grid({
  dataSource: data,
  allowFiltering: true,
  filterSettings: { type: 'Menu' },
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
    { field: 'ShipCity', headerText: 'Ship City', width: 100 },
    { field: 'ShipName', headerText: 'Ship Name', width: 100 }
  ],
  height: 273
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
```

```

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

* [allowFiltering](#) must be set as true to enable filter menu.

* Setting [columns.allowFiltering](#) as false will prevent filter menu rendering for a particular column.

Custom component in filter menu

The [column.filter.ui](#) is used to add custom filter components to a particular column.

To implement custom filter ui, define the following functions:

- **create**: Creates custom component.
- **write**: Wire events for custom component.
- **read**: Read the filter value from custom component.

In the following sample, dropdown is used as custom component in the OrderID column.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Filter);
var grid = new ej.grids.Grid({
  dataSource: data,
  allowFiltering: true,
  filterSettings: {type: 'Menu'},
  height: 273,
  columns: [
    {
      field: 'OrderID', headerText: 'Order ID', width: 120,
      textAlign: 'Right', filter: {
        ui: {
          create: function(args){
            var db = new ej.data.DataManager(data);
            var flValInput = new
ej.base.createElement('input', { className: 'flm-input' });
            args.target.appendChild(flValInput);
            this.dropInstance = new
ej.dropdowns.DropDownList({
              dataSource: new ej.data.DataManager(data),
              fields: { text: 'OrderID', value: 'OrderID'
},
              placeholder: 'Select a value',
              popupHeight: '200px'
            });
            this.dropInstance.appendTo(flValInput);
          },
          write: function(args){
            this.dropInstance.value = args.filteredValue;
          },
          read: function(args) {
            args.fltrObj.filterByColumn(args.column.field,
args.operator, this.dropInstance.value);
          }
        }
      }
    },
    { field: 'CustomerID', headerText: 'Customer Name', width: 150
},
    { field: 'OrderDate', headerText: 'Order Date', width: 130,
format: 'yMd', textAlign: 'Right' },
    { field: 'Freight', width: 120, format: 'C2', textAlign: 'Right'
}
  ]
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {

```

```

    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customizing filter menu operators list

You can customize the default filter operator list by defining the [filterSettings.operators](#) property.

The available options are:

- **stringOperator** - Defines the customized string operator list.
- **numberOperator** - Defines the customized number operator list.
- **dateOperator** - Defines the customized date operator list.
- **booleanOperator** - Defines the customized Boolean operator list.

The following sample illustrates customizing the string filter operators.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Filter);
var grid = new ej.grids.Grid({
    dataSource: data,
    allowFiltering: true,
    filterSettings: {type: 'Menu',
        operators: {
            stringOperator: [
                { value: 'startsWith', text: 'starts with' },
                { value: 'endsWith', text: 'ends with' },
                { value: 'contains', text: 'contains' }],
        }
    },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
        { field: 'ShipCity', headerText: 'Ship City', width: 100 },
        { field: 'ShipName', headerText: 'Ship Name', width: 100 }
    ],
    height: 273
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}

```

```
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Enable different filter for a column

You can use both **Menu** and **CheckBox** filter in a same Grid. To do so, set the [column.filter.type](#) as **Menu** or **CheckBox**.

In the following sample menu filter is enabled by default and checkbox filter is enabled for the CustomerID column using the [column.filter.type](#).

INDEX.JS

```
ej.grids.Grid.Inject(ej.grids.Filter);
var grid = new ej.grids.Grid({
  dataSource: data,
  allowFiltering: true,
  filterSettings: { type: 'Menu' },
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 120 ,
filter: { type : 'CheckBox' } },
    { field: 'ShipCity', headerText: 'Ship City', width: 100 },
    { field: 'ShipName', headerText: 'Ship Name', width: 100 }
  ],
  height: 273
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Filter by multiple keywords using filter menu

By default, the filtering action is performed based on the single keyword filter value from the built-in component of the filter menu dialog. Now data grid has an option to perform filtering actions based on multiple keywords instead of a single keyword alone. For this, set [filterSettings.type](#) as `Menu`.

In the following sample, filtering action with multiple keywords can be done by rendering the `MultiSelect` component as custom component in the OrderID column filter menu dialog.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Filter);
var grid = new ej.grids.Grid({
  dataSource: data,
  allowPaging: true,
  allowFiltering: true,
  filterSettings: { type: 'Menu' },
  columns: [
    {
      field: 'OrderID',
      headerText: 'Order ID',
      width: 120,
      textAlign: 'Right',
      filter: {
        ui: {
          create: function (args) {
            var dp = new ej.data.DataManager(data);
            var input = new ej.base.createElement('input', {
              className: 'flm-input',
            });
            args.target.appendChild(input);
            dropInstance = new ej.dropdowns.MultiSelect({
              dataSource: new ej.data.DataManager(data),
              fields: { text: 'OrderID', value: 'OrderID' },
              placeholder: 'Select a value',
              popupHeight: '200px',
              allowFiltering: true,
              mode: 'CheckBox',
            });
            dropInstance.appendTo(input);
          },
          write: function (args) {
            var grid = document.getElementById('Grid').ej2_instances[0];
            var filteredValue = [];
            grid.filterSettings.columns.map((item) => {
              if (item.field === 'OrderID' && item.value) {
                filteredValue.push(item.value);
              }
            });
            if (filteredValue.length > 0) {
              dropInstance.value = filteredValue;
            }
          },
          read: function (args) {
            var grid = document.getElementById('Grid').ej2_instances[0];
            grid.removeFilteredColsByField(args.column.field);
            args.fltrObj.filterByColumn(
              args.column.field,
              'contains',
              dropInstance.value
            );
          },
        },
      },
    },
    { field: 'CustomerID', headerText: 'Customer Name', width: 150 },
  ],
});

```

```

        {
            field: 'OrderDate',
            headerText: 'Order Date',
            width: 130,
            format: { type: 'dateTime', format: 'M/d/y hh:mm a' },
            textAlign: 'Right',
        },
        { field: 'Freight', width: 110, format: 'C2', textAlign: 'Right' },
    ],
    pageSettings: { pageCount: 5 },
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
      background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
      background-color: #eee;
    }
  </style>

```

```

        .e-expand::before {
            content: '\e5b8';
        }
        .empImage {
            margin: 6px 16px;
            float: left;
            width: 50px;
            height: 50px;
        }
    </style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Add current selection to filter checkbox

By default, the CheckBox filter can only filter the selected items. If filtering is done multiple times on the same column, the previously filtered values in the column will be cleared. Now, it is possible to retain those previous values by using the **Add current selection to filter** checkbox. This checkbox is displayed when data is searched in the search bar of the CheckBox filter.

The following image describes the above mentioned behavior.

Order ID	Customer Name	Order Date	Freight
10248		4/1996 10:10 AM	\$32.38
10249		5/1996 12:20 PM	\$11.61
10250		3/1996 08:40 AM	\$65.83
10251		3/1996 07:50 AM	\$41.34
10252		9/1996 12:05 PM	\$51.30
10253		10/1996 11:20 AM	\$58.17
10254		1/1996 10:00 AM	\$22.98
10255		2/1996 10:40 AM	\$148.33
10256		5/1996 08:50 PM	\$13.97
10257	Carlos Hernández	7/16/1996 03:20 AM	\$81.91
10258	Roland Mendel	7/17/1996 06:30 PM	\$140.51
10259	Francisco Chang	7/18/1996 01:20 AM	\$3.25

☒ Select All
 ☐ Add current selection to filter
 ☒ 10248

Filter Clear

<< < 1 2 3 4 5 ... > >>
 1 of 70 pages (830 items)

See also

- [How to perform filter by using Wildcard and LIKE operator filter](#)

Excel like filter in ##Platform_Name## Grid control

You can enable Excel like filter by defining. [filterSettings.type](#) as Excel. The excel menu contains an option such as Sorting, Clear filter, Sub menu for advanced filtering.

INDEX.JS

```
ej.grids.Grid.Inject(ej.grids.Filter);
var grid = new ej.grids.Grid({
  dataSource: data,
  allowFiltering: true,
  filterSettings: { type: 'Excel' },
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
    { field: 'ShipCity', headerText: 'Ship City', width: 100 },
    { field: 'ShipName', headerText: 'Ship Name', width: 100 }
  ],
});
```

```

        height: 273
    });
    grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
      background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
      background-color: #eee;
    }
    .e-expand::before {
      content: '\e5b8';
    }
    .empImage {
      margin: 6px 16px;
      float: left;
      width: 50px;
      height: 50px;
    }
  </style>

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

* By default, while opening the excel/checkbox filter in the Grid, the filter dialog will get and display the distinct data from the first 1000 records bound to the Grid to optimize performance. The remaining records will be returned as a result of the search option of the filter dialog.

* However, we can increase the excel/checkbox filter count by modifying the `filterChoiceCount` argument value(as per our need) in the `actionBegin` event when the `requestType` is `filterchoicerequest` or `filtersearchbegin`. This is demonstrated in the below code snippet,

```

`ts
function actionBegin(args: FilterSearchBeginEventArgs) {
if (args.requestType === "filterchoicerequest" || args.requestType === "filtersearchbegin") {
args.filterChoiceCount = 3000;
}
}
`

```

Render checkbox list data in on-demand for excel/checkbox filtering

The Excel/Checkbox filter type of Grid has a restriction where only the first 1000 unique sorted items are accessible to view in the filter dialog checkbox list content by scrolling. This limitation is in place to avoid any rendering delays when opening the filter dialog. However, the searching and filtering processes consider all unique items in that particular column.

The Excel/Checkbox filter in the Grid provides an option to load large data sets on-demand during scrolling to improve scrolling limitation functionality. This is achieved by setting the `filterSettings.enableInfiniteScrolling` property to **true**. This feature proves especially beneficial for managing extensive datasets, enhancing data loading performance in the checkbox list, and allowing interactive checkbox selection with persistence for the selection based on filtering criteria.

The Excel/Checkbox filter retrieves distinct data in ascending order, governed by its `filterSettings.itemsCount` property, with a default value of **50**. As the checkbox list data scroller reaches

its end, the next dataset is fetched and displayed, with the notable advantage that this process only requests new checkbox list data without redundantly fetching the existing loaded dataset.

Customize the items count for initial rendering

Based on the items count value, the Excel/Checkbox filter gets unique data and displayed in Excel/Checkbox filter content dialog. You can customize the count of on-demand data rendering for Excel/Checkbox filter by adjusting the [filterSettings.itemsCount](#) property. The default value is **50**

```
`ts
```

```
grid.filterSettings = { enableInfiniteScrolling = true, itemsCount = 40 };
```

```
,
```

It is recommended to keep the itemsCount below **300**. Higher values will result in unwanted whitespace because of DOM maintenance performance degradation.

Customize the loading animation effect

A loading effect is presented to signify that loading is in progress when the checkbox list data scroller reaches the end, and there is a delay in receiving the data response from the server. The loading effect during on-demand data retrieval for Excel/Checkbox filter can be customized using the [filterSettings.loadingIndicator](#) property. The default value is **Shimmer**.

```
`ts
```

```
grid.filterSettings = { enableInfiniteScrolling = true, loadingIndicator = 'Spinner' };
```

```
,
```

In the provided example, On-Demand Excel filter has been enabled for the ##Platform_Name## Grid

INDEX.JS

```
ej.grids.Grid.Inject(ej.grids.Filter, ej.grids.Page, ej.grids.Selection,
ej.grids.Sort);
var urlapi = new ej.data.DataManager({
  url: 'https://services.syncfusion.com/js/production/api/UrlDataSource',
  adaptor: new ej.data.UrlAdaptor()
});
var grid = new ej.grids.Grid({
  dataSource: urlapi,
  query: new ej.data.Query().addParams('dataCount', '10000'),
  allowPaging: true,
  allowFiltering: true,
  allowSorting: true,
  filterSettings: { type: 'Excel', enableInfiniteScrolling: true,
itemsCount: 40, loadingIndicator: 'Spinner' },
  columns: [
    { field: 'EmployeeID', headerText: 'Employee ID', isPrimaryKey:
true, width: '120' },
    { field: 'Employees', headerText: 'Employee Name', width: '150' },
    { field: 'Designation', headerText: 'Designation', width: '130' },
    {
      field: 'CurrentSalary', headerText: 'Current Salary', format:
'C2',
      textAlign: 'Right', width: '120'
    },
  ],
},
```

```

    pageSettings: { pageCount: 5 }
  });
  grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Grid"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

See also

- [How to perform filter by using Wildcard and LIKE operator filter](#)

Searching in ##Platform_Name## Grid control

You can search records in a Grid, by using the [search](#) method with search key as a parameter. This also provides an option to integrate search text box in grid's toolbar by adding [search](#) item to the [toolbar](#).

To search records, inject the [Search](#) module in the grid.

The clear icon is shown in the Data Grid search text box when it is focused on search text or after typing the single character in the search text box. A single click of the clear icon clears the text in the search box as well as the search results in the Data Grid.

INDEX.JS

```
ej.grids.Grid.Inject(ej.grids.Toolbar);
var grid = new ej.grids.Grid({
  dataSource: data,
  toolbar: ['Search'],
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
    { field: 'CustomerID', width: 150, headerText: 'Customer ID', type:
'string' },
    { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C' },
    { field: 'OrderDate', headerText: 'Order Date', textAlign: 'Right',
width: 140, format: 'yMd' }
  ],
  height: 272
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Initial search

To apply search at initial rendering, set the fields, operator, key, and ignoreCase in the [searchSettings](#).

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Toolbar);
var grid = new ej.grids.Grid({
    dataSource: data,
    toolbar: ['Search'],

```

```

searchSettings: { fields: ['CustomerID'], operator: 'contains', key:
'Ha', ignoreCase: true },
columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
    { field: 'ShipCity', headerText: 'Ship City', width: 150 },
    { field: 'ShipName', headerText: 'Ship Name', width: 150 }
],
height: 272
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
<title>EJ2 Grid</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
.e-row[aria-selected="true"] .e-customizedExpandcell {
    background-color: #e0e0e0;
}
.e-grid.e-gridhover tr[role='row']:hover {
    background-color: #eee;
}
.e-expand::before {

```

```

        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

By default, grid searches all the bound column values. To customize this behavior define the [searchSettings.fields](#) property.

Search operators

The search operator can be defined in the [searchSettings.operator](#) property to configure specific searching.

The following operators are supported in searching:

Operator | Description

startsWith | Checks whether a value begins with the specified value.

endsWith | Checks whether a value ends with the specified value.

contains | Checks whether a value contains the specified value.

equal | Checks whether a value is equal to the specified value.

notEqual | Checks for values not equal to the specified value.

By default, the [searchSettings.operator](#) value is `contains`.

Search by external button

To search grid records from an external button, invoke the [search](#) method.

INDEX.JS

```
var grid = new ej.grids.Grid({
```

```

        dataSource: data,
        columns: [
            { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
            { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
            { field: 'ShipCity', headerText: 'Ship City', width: 150 },
            { field: 'ShipName', headerText: 'Ship Name', width: 150 }
        ],
        height: 260
    });
    grid.appendTo('#Grid');
    var searchBtn= new ej.buttons.Button();
    searchBtn.appendTo('#search');
    document.getElementById('search').addEventListener('click', function() {
        var searchText = document.getElementsByClassName('searchtext')[0].value;
        grid.search(searchText);
    });

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>

```

```

<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="e-float-input" style="width: 200px; display: inline-block;">
            <input type="text" class="searchtext">
            <span class="e-float-line"></span>
            <label class="e-float-text">Search text</label>
        </div>
        <button id="search">Search</button>
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Search specific columns

By default, grid searches all visible columns. You can search specific columns by defining the specific column's field names in the [searchSettings.fields](#) property.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Toolbar);
var grid = new ej.grids.Grid({
    dataSource: data,
    searchSettings: { fields: ['CustomerID', 'ShipCity', 'ShipName'] },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right', width: 100 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
        { field: 'ShipCity', headerText: 'Ship City', width: 100 },
        { field: 'ShipName', headerText: 'Ship Name', width: 100 }
    ],
    toolbar: ['Search'],
    height: 270
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">

```

```

<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Clear search by external button

To clear the searched grid records from the external button, set [searchSettings.key](#) property as empty string.

INDEX.JS

```

var grid = new ej.grids.Grid({
    dataSource: data,

```

```

        toolbar: ['Search'],
        searchSettings: { fields: ['CustomerID'], operator: 'contains', key:
'Ha', ignoreCase: true },
        columns: [
            { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
            { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
            { field: 'ShipCity', headerText: 'Ship City', width: 150 },
            { field: 'ShipName', headerText: 'Ship Name', width: 150 }
        ],
        height: 272
    });
    grid.appendTo('#Grid');
    let clearBtn = new ej.buttons.Button();
    clearBtn.appendTo('#clear');
    document.getElementById('clear').addEventListener('click', () => {
        grid.searchSettings.key='';
    });

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

```



```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="clear">Clear Search</button>
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Search on each key stroke

You can search the Grid data on each key stroke by binding the `keyup` event for the search input element inside the `created` event. Inside the `keyup` handler you can search the Grid by invoking the `search` method of the Grid component.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Toolbar);
var grid = new ej.grids.Grid({
    dataSource: data,
    toolbar: ['Search'],
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'ShipCity', headerText: 'Ship City', width: 150 },
        { field: 'ShipName', headerText: 'Ship Name', width: 150 }
    ],
    height: 272,
    created: () => {
        document.getElementById(grid.element.id +
"_searchbar").addEventListener('keyup', () => {
            grid.search(event.target.value)
        });
    }
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

```

```

<meta name="description" content="Typescript Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Highlight the search text

You can highlight the search text in the Grid content by adding the style inside the [queryCellInfo](#) event. you can get the search keyword from the [actionBegin](#) event.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Toolbar);
var key = "";
var grid = new ej.grids.Grid({
    dataSource: data,
    toolbar: ['Search'],
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'ShipCity', headerText: 'Ship City', width: 150 },
        { field: 'ShipName', headerText: 'Ship Name', width: 150 }
    ],
    height: 272,
    actionBegin: args => {
        if (args.requestType === 'searching') {
            key = args.searchString.toLowerCase();
        }
    },
    queryCellInfo: args => {
        if (key !== '') {
            var cellContent = args.data[args.column.field];
            var parsedContent = cellContent.toString().toLowerCase();
            if (parsedContent.includes(key.toLowerCase())) {
                var i = 0;
                var searchStr = '';
                while (i < key.length) {
                    var index = parsedContent.indexOf(key[i]);
                    searchStr = searchStr + cellContent.toString()[index];
                    i++;
                }
                args.cell.innerHTML = args.cell.innerText.replaceAll(
                    searchStr,
                    "<span class='customcss'>" + searchStr + "</span>"
                );
            }
        }
    }
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Perform search operation in Grid using multiple keywords

You can perform a searching operation in the Grid using multiple keywords. This can be achieved by the [actionBegin](#) event of the Grid. In the following sample, we have performed the searching with multiple keywords by using the query property of grid when the requestType is searching in the [actionBegin](#) event.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Toolbar);
var values;
var key = '';
var refresh = false;
var removeQuery = false;
var valueAssign = false;
var grid = new ej.grids.Grid({

```

```

        dataSource: data,
        toolbar: ['Search'],
        columns: [
            {
                field: 'OrderID', headerText: 'Order ID', textAlign:
'Center', width: 120,
            },
            {
                field: 'CustomerID', headerText: 'Customer ID', textAlign:
'Center', width: 150,
            },
            {
                field: 'ShipCity', headerText: 'Ship City', textAlign:
'Center', width: 150,
            },
            {
                field: 'ShipName', headerText: 'Ship Name', textAlign:
'Center', width: 150,
            },
        ],
        searchSettings: {
            fields: ['CustomerID', 'OrderID', 'ShipCity', 'ShipName'],
            key: '',
        },
        height: 272,
        actionBegin: (args) => {
            if (args.requestType == 'searching') {
                const keys = args.searchString.split(',');
                var flag = true;
                var predicate;
                if (keys.length > 1) {
                    if (grid.searchSettings.key !== '') {
                        values = args.searchString;
                        keys.forEach((key) => {
                            grid.getColumns().forEach((col) => {
                                if (flag) {
                                    predicate = new
ej.data.Predicate(col.field, 'contains', key, true);
                                    flag = false;
                                }
                                else {
                                    var predic = new
ej.data.Predicate(col.field, 'contains', key, true);
                                    predicate = predicate.or(predic);
                                }
                            });
                        });
                    }
                    grid.query = new ej.data.Query().where(predicate);
                    grid.searchSettings.key = '';
                    refresh = true;
                    valueAssign = true;
                    removeQuery = true;
                    grid.refresh();
                }
            }
        },
    },

```

```

    actionComplete: (args) => {
        if (args.requestType === 'refresh' && valueAssign) {
            document.getElementById(grid.element.id + '_searchbar').value =
values;
            valueAssign = false;
        }
        else if (
            document.getElementById(grid.element.id + '_searchbar').value
=== '' &&
            args.requestType === 'refresh' &&
            removeQuery
        )
        {
            grid.query = new ej.data.Query();
            removeQuery = false;
            grid.refresh();
        }
    },
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

* Search operation can be performed in foreign key column based on following way.

* When a value is searched on a grid with the foreign key column, a filter query is sent to the foreign key data source, and the appropriate column is filtered depending on the search value. The search query will be sent to the grid data source, and the value of the associated field will be returned.

See Also

- [How to perform searching in Date type column](#)
- [How to search the records in grid on each keystroke](#)
- [How to perform search by using Wildcard and LIKE operator filter](#)

Paging in ##Platform_Name## Grid control

Paging provides an option to display Grid data in page segments. To enable paging, set the [allowPaging](#) to true. When paging is enabled, pager component renders at the bottom of the grid. Paging options can be configured through the [pageSettings](#).

In the below sample, `pageSize` is calculated based on the grid height by using the `load` event.

To use paging, inject the [Page](#) module in the grid.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Page);
var grid = new ej.grids.Grid({
    dataSource: data,
    allowPaging: true,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'ShipCity', headerText: 'Ship City', width: 150 },
        { field: 'ShipName', headerText: 'Ship Name', width: 150 }
    ],

```

```

height: 325,
load: () => {
    var rowHeight = grid.getRowHeight(); //height of the each row
    var gridHeight = grid.height;
    var pageSize = grid.pageSettings.pageSize; //initial page size
    var pageResize = (gridHeight - (pageSize * rowHeight)) / rowHeight;
    //new page size is obtained here
    grid.pageSettings.pageSize = pageSize + Math.round(pageResize);
}
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <style>
        .e-row[aria-selected="true"] .e-customizedExpandcell {
            background-color: #e0e0e0;
        }
        .e-grid.e-gridhover tr[role='row']:hover {
            background-color: #eee;
        }
        .e-expand::before {
            content: '\e5b8';
        }
    </style>

```



```

    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

You can achieve better performance by using grid paging to fetch only a pre-defined number of records from the data source.

Template

You can use custom elements inside the pager instead of default elements. The custom elements can be defined by using the [template](#) property. Inside this template, you can access the [CurrentPage](#), [pageSize](#), [pageCount](#), [totalPage](#) and [totalRecordCount](#) values.

INDEX.JS

```

var updateTemplate = function() {
    this.numeric = new ej.inputs.NumericTextBox({
        min: 1,
        max: 3,
        step: 1,
        format: '###.##',
        change: (args) => {
            var value = args.value;
            grid.goToPage(value);
        }
    });
    this.numeric.appendTo('#currentPage');
};
var flag = true;
ej.grids.Grid.Inject(ej.grids.Page);
var grid = new ej.grids.Grid({
    dataSource: data,
    allowPaging: true,

```

```

columns: [
    { field: 'OrderID', headerText: 'Order ID', width: 120 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
    { field: 'ShipCity', headerText: 'Ship City', width: 150 },
    { field: 'ShipName', headerText: 'Ship Name', width: 150 }
],
pageSettings: { template: '#template', pageSize: 7 },
dataBound: function() {
    if (flag) {
        flag = false;
        updateTemplate();
    }
},
actionComplete: (args) => {
    if (args.requestType === 'paging') {
        updateTemplate();
    }
}
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

```

```

<script id="template" type="text/x-template">
  <div class="e-pagertemplate">
    <div class="col-lg-12 control-section">
      <div class="content-wrapper">
        <input id="currentPage" type="text" value=${currentPage}>
      </div>
    </div>

    <div id="totalPages" class="e-pagertemplatemessage" style="margin-top:5px;margin-left:30px;border: none; display: inline-block ">
      <span class="e-pagenomsg">${currentPage} of ${totalPages} pages (${{totalRecordsCount} items}</span>
    </div>
  </div>
</script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js" type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Grid"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Pager with Page Size Dropdown

The pager Dropdown allows you to change the number of records in the Grid dynamically. It can be enabled by defining the `pageSettings.pageSize` property as true.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Page);
var grid = new ej.grids.Grid({
  dataSource: data,
  allowPaging: true,
  pageSettings: { pageSize: true, pageSize: 8 },
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right', width: 120 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
    { field: 'ShipCity', headerText: 'Ship City', width: 150 },
    { field: 'ShipName', headerText: 'Ship Name', width: 150 }
  ]
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
      background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
      background-color: #eee;
    }
    .e-expand::before {
      content: '\e5b8';
    }
    .empImage {
      margin: 6px 16px;
      float: left;
      width: 50px;
      height: 50px;
    }
  </style>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>

```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

How to render Pager at the Top of the Grid

By default, Pager will be rendered at the bottom of the Grid. You can also render the Pager at the top of the Grid by using the `dataBound` event.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Page, ej.grids.Toolbar);
var initialGridLoad = true;
var grid = new ej.grids.Grid({
    dataSource: data,
    allowPaging: true,
    toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
    pageSettings: { pageSizes: true, pageSize: 8 },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'ShipCity', headerText: 'Ship City', width: 150 },
        { field: 'ShipName', headerText: 'Ship Name', width: 150 }
    ]
});
grid.appendTo('#Grid');
grid.dataBound = () =>{
    if (initialGridLoad) {
        initialGridLoad = false;
        var pager = document.getElementsByClassName('e-gridpager');
        var topElement;
        if (grid.allowGrouping || grid.toolbar) {
            topElement = grid.allowGrouping ?
document.getElementsByClassName('e-groupdroparea') :
document.getElementsByClassName('e-toolbar');
        } else {
            topElement = document.getElementsByClassName('e-gridheader');
        }
        grid.element.insertBefore(pager[0], topElement[0]);
    }
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
      background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
      background-color: #eee;
    }
    .e-expand::before {
      content: '\e5b8';
    }
    .empImage {
      margin: 6px 16px;
      float: left;
      width: 50px;
      height: 50px;
    }
  </style>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>

```

```

</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

During the paging action, the pager component triggers the below three events.

- * The **created** event triggers when Pager is created.
- * The **click** event triggers when the numeric items in the pager is clicked.
- * The **dropDownChanged** event triggers when pageSize DropDownList value is selected.

See Also

- [Group with Paging](#)

Scrolling in ##Platform_Name## Grid control

The scrollbar will be displayed in the grid when content exceeds the element [width](#) or [height](#). The vertical and horizontal scrollbars will be displayed based on the following criteria:

- The vertical scrollbar appears when the total height of rows present in the grid exceeds its element height.
- The horizontal scrollbar appears when the sum of columns width exceeds the grid element width.
- The [height](#) and [width](#) are used to set the grid height and width, respectively.

The default value for [height](#) and [width](#) is **auto**.

Set width and height

To specify the [width](#) and [height](#) of the scroller in the pixel, set the pixel value to a number.

INDEX.JS

```

var grid = new ej.grids.Grid({
    dataSource: data,
    height: 315,
    width: 400,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'EmployeeID', headerText: 'Employee ID', textAlign:
'Right', width: 120 },

```

```

        { field: 'ShipCity', headerText: 'Ship City', width: 150 },
        { field: 'ShipCountry', headerText: 'Ship Country', width: 150 },
        { field: 'ShipName', headerText: 'Ship Name', width: 150 }
    ]
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="height:350px;">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');

```



```

if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Responsive with parent container

Specify the [width](#) and [height](#) as **100%** to make the grid element fill its parent container.

Setting the [height](#) to **100%** requires the grid parent element to have explicit height.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Page, ej.grids.Group, ej.grids.Sort,
ej.grids.Filter);
var grid = new ej.grids.Grid({
    dataSource: data,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign:
'Right', width: 120, type: 'number' },
        { field: 'CustomerID', width: 140, headerText: 'Customer
ID', type: 'string' },
        { field: 'Freight', headerText: 'Freight', textAlign:
'Right', width: 120, format: 'C' },
        { field: 'OrderDate', headerText: 'Order Date', width: 140,
format: 'yMd' }
    ],
    height: '100%',
    allowGrouping: true,
    allowPaging: true,
    allowSorting: true,
    allowFiltering: true
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-resizable {
        resize: both;
        overflow: auto;
        border: 1px solid red;
        padding: 10px;
        height: 300px;
        min-height: 250px;
        min-width: 250px;
    }
    .e-text{
        font-family: Helvetica, sans-serif;
        font-size: 14px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <p class="e-text"> The parent container can be resizable by dragging the
bottom-right corner.</p>
    <div id="container" class="e-resizable">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Sticky Header

You can make the Grid column headers remain fixed while scrolling by using the [enableStickyHeader](#) property.

In the below demo, the Grid headers will be sticky while scrolling the Grid's parent div element.

INDEX.JS

```
var grid = new ej.grids.Grid({
  dataSource: data,
  enableStickyHeader: true,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
    { field: 'EmployeeID', headerText: 'Employee ID', textAlign:
'Right', width: 120 },
    { field: 'ShipName', headerText: 'Ship Name', width: 150 }
  ]
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="height:350px;">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Scroll to selected row

You can scroll the grid content to the selected row position by using the [rowSelected](#) event.

INDEX.JS

```

var grid = new ej.grids.Grid({
    dataSource: data,
    height: '270',
    width: '100%',
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'EmployeeID', headerText: 'Employee ID', textAlign:
'Right', width: 120 },
        { field: 'ShipName', headerText: 'Ship Name', width: 150 }
    ],
    rowSelected: rowSelected
});
grid.appendTo('#Grid');
var numeric = new ej.inputs.NumericTextBox({
    width: 200,
    min: 0,
    showSpinButton: false,
    format: 'N',
    placeholder: 'Enter index to select a row',
    change: onchange
}, '#numeric');
function onchange() {
    grid.selectionModule.selectRow(parseInt(numeric.getText(), 10));
}
function rowSelected(args) {
    var rowHeight =
grid.getRows()[grid.getSelectedRowIndex()[0]].scrollHeight;

```

```
grid.getContent().children[0].scrollTop = rowHeight *
grid.getSelectedRowIndexes()[0];
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="height:350px;">
    <input id="numeric" type="text">
    <div id="Grid"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
```

```

}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Hide the scrollbar when the content is not overflown

You can hide the scrollbar of Grid content by using the [hideScroll](#) method when the content doesn't overflow its parent element.

In the following sample, we have invoked the [hideScroll](#) method inside the [dataBound](#) event.

INDEX.JS

```

var grid = new ej.grids.Grid({
  dataSource: data.slice(0, 5),
  height: '315',
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
    { field: 'EmployeeID', headerText: 'Employee ID', textAlign:
'Right', width: 120 },
    { field: 'ShipName', headerText: 'Ship Name', width: 150 }
  ],
  dataBound: () => {
    grid.hideScroll();
  }
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="height:350px;">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Frozen in ##Platform_Name## Grid control

Frozen rows and columns provides an option to make rows and columns always visible in the top and left side of the grid while scrolling.

In this demo, the [frozenColumns](#) is set as '2' and the [frozenRows](#) is set as '3'. Hence, the left two columns and top three rows are frozen.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Freeze);
var grid = new ej.grids.Grid({
    dataSource: data,
    height: 315,
    allowSelection: false,
    enableHover: false,
    frozenRows: 3,
    frozenColumns: 2,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'OrderDate', headerText: 'Order Date', width: 130, format:
'yMd', textAlign: 'Right' },
        { field: 'EmployeeID', headerText: 'Employee ID', textAlign:
'Right', width: 120 },

```

```

        { field: 'ShipName', headerText: 'Ship Name', width: 150 },
        { field: 'ShipAddress', headerText: 'Ship Address', width: 170 },
        { field: 'ShipCity', headerText: 'Ship City', width: 150 },
        { field: 'ShipCountry', headerText: 'Ship Country', width: 150 },
        { field: 'ShipRegion', headerText: 'Ship Region', width: 150 },
        { field: 'ShipPostalCode', headerText: 'Ship Postal Code', width:
150 },
        { field: 'Freight', headerText: 'Freight', width: 120 }
    ]
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```



```

<div id="container" style="height:350px;">
  <div id="Grid"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

* Frozen rows and columns should not be set outside the grid view port.

* Frozen Grid will support row and column virtualization feature, which helps to improve the Grid performance while loading a large dataset.

Limitations of Frozen Grid

The following features are not supported in frozen rows and columns:

- Detail Template
- Hierarchy Grid

Freeze particular columns

You can use [isFrozen](#) property to freeze selected columns in grid.

In this demo, the columns with field name `OrderID` and `EmployeeID` is frozen using the `isFrozen` property.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Freeze);
var grid = new ej.grids.Grid({
  dataSource: data,
  height: 315,
  allowSelection: false,
  enableHover: false,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, isFrozen: true },
    { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
    { field: 'OrderDate', headerText: 'Order Date', width: 130, format:
'yMd', textAlign: 'Right' },
    { field: 'EmployeeID', headerText: 'Employee ID', textAlign:
'Right', width: 120, isFrozen: true },
    { field: 'ShipName', headerText: 'Ship Name', width: 150 },
    { field: 'ShipAddress', headerText: 'Ship Address', width: 170 },
    { field: 'ShipCity', headerText: 'Ship City', width: 150 },
    { field: 'ShipCountry', headerText: 'Ship Country', width: 150 },
    { field: 'ShipRegion', headerText: 'Ship Region', width: 150 },
    { field: 'ShipPostalCode', headerText: 'Ship Postal Code', width:
150 },
    { field: 'Freight', headerText: 'Freight', width: 120 }
  ]
});

```

```
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="height:350px;">
    <div id="Grid"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
<script src="index.js" type="text/javascript"></script>
```

```
</body></html>
```

* [isFrozen](#) is not compatible with the Freeze direction feature.

Freeze Direction

You can freeze the Grid columns on the left or right side by using the [column.freeze](#) property and the remaining columns will be movable. The grid will automatically move the columns to the left or right position based on the [column.freeze](#) value.

Types of the [column.freeze](#) directions:

- **Left:** Allows you to freeze the columns at the left.
- **Right:** Allows you to freeze the columns at the right.
- **Fixed:** Allows you to lock the column at a fixed position by ensuring its visibility during horizontal scroll.

In this demo, the **ShipCountry** column is frozen at the left and the **CustomerID** column is frozen at the right side of the content table.

INDEX.JS

```
ej.grids.Grid.Inject(ej.grids.Freeze);
var grid = new ej.grids.Grid({
  dataSource: data,
  height: 315,
  enableHover: false,
  frozenRows: 2,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
    { field: 'Freight', headerText: 'Freight', format: 'C2', width: 120
},
    { field: 'CustomerID', headerText: 'Customer ID', width: 150,
freeze: 'Right' },
    { field: 'OrderDate', headerText: 'Order Date', width: 130, format:
'yMd', textAlign: 'Right' },
    { field: 'ShipName', headerText: 'Ship Name', width: 150 },
    { field: 'ShipAddress', headerText: 'Ship Address', width: 170 },
    { field: 'ShipCity', headerText: 'Ship City', width: 150 },
    { field: 'ShipCountry', headerText: 'Ship Country', width: 150,
freeze: 'Left' }
  ]
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="height:350px;">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

* Freeze Direction is not compatible with the [isFrozen](#) and [frozenColumns](#) properties.

Deprecated Methods

Deprecated Methods | [Previous](#) | [Current](#) | [Suggested Alternative Methods](#) | [Example for achieving the same results](#)

getMovableRows() | In the previous architecture of frozen grid, three separate tables were created for the left, right, and movable contents. When calling this method, it would return only the movable table

rows (tr's). | In the current architecture, the frozen left, right, and movable sections are applied within a single table. When calling this method, it will return all table rows (tr's) of the entire table. However, in this approach, we have introduced the `e-unfreeze` class for movable cells. This allows us to selectively retrieve the movable rows using the `e-unfreeze` class selector. | `getRows()` | `gridInstance.getMovableRows()[0].querySelectorAll('.e-unfreeze')` // Deprecated

 (or)

 `gridInstance.getRows()[0].querySelectorAll('.e-unfreeze')` // Alternative method

`getFrozenRightRows()` | In the previous architecture, this method would return only the table rows (tr's) from the freeze right table. | In the current architecture, the frozen left, right, and movable sections are applied within a single table. When calling this method, it will return all the rows (tr's) of the entire table. In this new approach, we have introduced the `e-rightfreeze` class for right freeze cells. As a result, you can now selectively retrieve the right freeze rows using the `e-rightfreeze` class selector. | `getRows()` | `gridInstance.getFrozenRightRows()[0].querySelectorAll('.e-rightfreeze')` // Deprecated

 (or)

 `gridInstance.getRows()[0].querySelectorAll('.e-rightfreeze')` // Alternative method

`getMovableRowByIndex()`
 `getFrozenRowByIndex()`
 `getFrozenRightRowByIndex()` | In the previous architecture, you could select rows by using separate methods for each table section. Like,
 `getMovableRowByIndex` - select a movable row
 `getFrozenRowByIndex` - select a freeze row
 `getFrozenRightRowByIndex` - select a right freeze row. | In the current architecture, the `getMovableRowByIndex`, `getFrozenRightRowByIndex` and `getFrozenRowByIndex` methods all return the same table row (tr) based on the given index. Additionally, class names for table cells (td's) have been separated as follows:
 Left-Freeze : `e-leftfreeze`
 Movable : `e-unfreeze`
 Right-Freeze : `e-rightfreeze`.
This separation of class names makes it easier to target and customize the cells within the particular row. | `getRowByIndex()` | **To get the left freeze cells:**
 `gridInstance.getRowByIndex(1).querySelectorAll('.e-leftfreeze')`

 To get the movable cells:
 `gridInstance.getRowByIndex(1).querySelectorAll('.e-unfreeze')`

 To get the right freeze cells:
 `gridInstance.getRowByIndex(1).querySelectorAll('.e-rightfreeze')`

`getMovableCellFromIndex()`
 `getFrozenRightCellFromIndex()` | `getMovableCellFromIndex()` - select a particular cell in the movable table.
 `getFrozenRightCellFromIndex()` - select a particular cell in the right freeze table. | In the new approach, you can select a particular cell by using both the `getFrozenRightCellFromIndex` and `getMovableCellFromIndex` methods. | `getCellFromIndex()` | `gridInstance.getCellFromIndex(1,1)`

`getMovableDataRows()`
 `getFrozenRightDataRows()`
 `getFrozenDataRows()` | These methods returns the viewport data rows for the freeze, movable, and right tables separately. | In the new approach, when calling the `getMovableDataRows`, `getFrozenRightDataRows`, and `getFrozenDataRows` methods, returns the entire viewport data rows. You can then select specific cells within these rows using the following selectors
 Left-Freeze : `e-leftfreeze`
 Movable : `e-unfreeze`
 * Right-Freeze : `e-rightfreeze`. | `getDataRows()` | **To get the movable data cells:**
 `gridInstance.getDataRows()[0].querySelectorAll('.e-unfreeze')`

 To get the right freeze data cells:
 `gridInstance.getDataRows()[0].querySelectorAll('.e-rightfreeze')`

 To get the left freeze data cells:
 `gridInstance.getDataRows()[0].querySelectorAll('.e-leftfreeze')`

`getMovableColumnHeaderByIndex()`
 `getFrozenRightColumnHeaderByIndex()`
 `getFrozenLeftColumnHeaderByIndex()` | In the previous architecture, these methods selects the movable, right freeze, and left freeze headers from the table separately. | In the new approach, when calling the `getMovableColumnHeaderByIndex`, `getFrozenRightColumnHeaderByIndex`, and

`getFrozenLeftColumnHeaderByIndex` methods, you will still receive the same results as before. | `getColumnHeaderByIndex()` | `gridInstance.getColumnHeaderByIndex(1)`

When a validation message is displayed in the frozen part (Left, Right, Fixed) of the table, scrolling is prevented until the validation message is cleared.

Virtual scroll in ##Platform_Name## Grid control

Grid allows you to load large amount of data without performance degradation.

To use virtualization, you need to inject `VirtualScroll` module in grid.

Row Virtualization

Row virtualization allows you to load and render rows only in the content viewport. It is an alternative way of paging in which the data will be loaded while scrolling vertically. To setup the row virtualization, you need to define [enableVirtualization](#) as true and content height by [height](#) property.

The number of records displayed in the Grid is determined implicitly by height of the content area. Also, you have an option to define a visible number of records by the [pageSettings.pageSize](#) property. The loaded data will be cached and reused when it is needed for next time.

INDEX.JS

```
var names = ['TOM', 'Hawk', 'Jon', 'Chandler', 'Monica', 'Rachel', 'Phoebe',
'Gunter', 'Ross', 'Geller', 'Joey', 'Bing', 'Tribbiani', 'Janice', 'Bong',
'Perk', 'Green', 'Ken', 'Adams'];
var hours = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
var designation = ['Manager', 'Engineer 1', 'Engineer 2', 'Developer',
'Tester'];
var statusValue = ['Completed', 'Open', 'In Progress', 'Review', 'Testing']
var data = function(count) {
    var result = [];
    for (var i = 0; i < count; i++) {
        result.push({
            TaskID: i + 1,
            Engineer: names[Math.round(Math.random() * names.length)] ||
names[0],
            Designation: designation[Math.round(Math.random() *
designation.length)] || designation[0],
            Estimation: hours[Math.round(Math.random() * hours.length)] ||
hours[0],
            Status: statusValue[Math.round(Math.random() *
statusValue.length)] || statusValue[0]
        });
    }
    return result;
};
(window).getStatus = function(statusValue){
    var colors = { 'Completed': 'green', 'Open': 'red', 'In Progress':
'#FB1E77', 'Review': 'brown', 'Testing': '#1EC1FB' };
    return '<span style="color:' + colors[statusValue] + '>' + statusValue
+ '</span>';
};
ej.grids.Grid.Inject(ej.grids.VirtualScroll, ej.grids.Edit,
ej.grids.Toolbar);
var grid = new ej.grids.Grid({
    dataSource: data(1000),
```

```

height: 300,
enableVirtualization: true,
pageSettings: { pageSize: 50 },
editSettings: { allowAdding: true, allowEditing: true, allowDeleting:
true, mode: 'Normal' },
toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
columns: [
    { field: 'TaskID', headerText: 'Task ID', textAlign: 'Right', width:
100, type: 'number', isPrimaryKey: true, validationRules: { required: true }
},
    { field: 'Engineer', width: 100 },
    { field: 'Designation', width: 140, editType: 'dropdownedit',
validationRules: { required: true } },
    { field: 'Estimation', textAlign: 'Right', width: 110, editType:
'numericedit', validationRules: { required: true } },
    { field: 'Status', width: 140, template:
'${getStatus(data.Status)}', editType: 'dropdownedit' }
]
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

```

```

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Column Virtualization

Column virtualization allows you to virtualize columns. It will render columns which are in the viewport. You can scroll horizontally to view more columns.

To setup the column virtualization, set the

[enableVirtualization](#) and

[enableColumnVirtualization](#) properties as `true`.

INDEX.JS

```

var virtualData = [];
var names = ['hardire', 'abramjo01', 'aubucch01', 'Hook', 'Rumpelstiltskin',
'Belle', 'Emma', 'Regina', 'Aurora', 'Elsa', 'Anna', 'Snow White', 'Prince
Charming', 'Cora', 'Zelena', 'August', 'Mulan', 'Graham', 'Discord', 'Will',
'Robin Hood', 'Jiminy Cricket', 'Henry', 'Neal', 'Red', 'Aaran', 'Aaren',
'Aarez', 'Aarman', 'Aaron', 'Aaron-James', 'Aarron', 'Aaryan', 'Aaryn',
'Aayan', 'Aazaan', 'Abaan', 'Abbas', 'Abdallah', 'Abdalroof', 'Abdihakim',
'Abdirahman', 'Abdisalam', 'Abdul', 'Abdul-Aziz', 'Abdulbasir',

```



```

'Abdulkadir', 'Abdulkarem', 'Abdulkhader', 'Abdullah', 'Abdul-Majeed',
'Abdulmalik', 'Abdul-Rehman', 'Abdur', 'Abdurraheem', 'Abdur-Rahman',
'Abdur-Rehmaan', 'Abel', 'Abhinav', 'Abhisumant', 'Abid', 'Abir', 'Abraham',
'Abu', 'Abubakar', 'Ace', 'Adain', 'Adam', 'Adam-James', 'Addison',
'Addisson', 'Adegbola', 'Adegbolahan', 'Aden', 'Adenn', 'Adie', 'Adil',
'Aditya', 'Adnan', 'Adrian', 'Adrien', 'Aedan', 'Aedin', 'Aedyn', 'Aeron',
'Afonso', 'Ahmad', 'Ahmed', 'Ahmed-Aziz', 'Ahoua', 'Ahtasham', 'Aiadan',
'Aidan', 'Aiden', 'Aiden-Jack', 'Aiden-Vee'];
function dataSource() {
    for (var i = 0; i < 1000; i++) {
        virtualData.push({
            'SNo': i + 1,
            'FIELD1': names[Math.floor(Math.random() * names.length)],
            'FIELD2': 1967 + (i % 10), 'FIELD3': Math.floor(Math.random() *
200),
            'FIELD4': Math.floor(Math.random() * 100), 'FIELD5':
Math.floor(Math.random() * 2000), 'FIELD6': Math.floor(Math.random() *
1000), 'FIELD7': Math.floor(Math.random() * 100), 'FIELD8':
Math.floor(Math.random() * 10), 'FIELD9': Math.floor(Math.random() * 10),
'FIELD10': Math.floor(Math.random() * 100), 'FIELD11':
Math.floor(Math.random() * 100), 'FIELD12': Math.floor(Math.random() *
1000), 'FIELD13': Math.floor(Math.random() * 10), 'FIELD14':
Math.floor(Math.random() * 10), 'FIELD15': Math.floor(Math.random() * 1000),
'FIELD16': Math.floor(Math.random() * 200), 'FIELD17':
Math.floor(Math.random() * 300), 'FIELD18': Math.floor(Math.random() * 400),
'FIELD19': Math.floor(Math.random() * 500), 'FIELD20':
Math.floor(Math.random() * 700), 'FIELD21': Math.floor(Math.random() * 800),
'FIELD22': Math.floor(Math.random() * 1000), 'FIELD23':
Math.floor(Math.random() * 2000), 'FIELD24': Math.floor(Math.random() *
150), 'FIELD25': Math.floor(Math.random() * 1000), 'FIELD26':
Math.floor(Math.random() * 100), 'FIELD27': Math.floor(Math.random() * 400),
'FIELD28': Math.floor(Math.random() * 600), 'FIELD29':
Math.floor(Math.random() * 500), 'FIELD30': Math.floor(Math.random() * 300),
        });
    }
}
dataSource();
ej.grids.Grid.Inject(ej.grids.VirtualScroll, ej.grids.Edit,
ej.grids.Toolbar);
var grid = new ej.grids.Grid({
    dataSource: virtualData,
    enableVirtualization: true,
    enableColumnVirtualization: true,
    height: 300,
    editSettings: { allowAdding: true, allowEditing: true, allowDeleting:
true, mode: 'Normal' },
    toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
    columns: [
        { field: 'SNo', headerText: 'S.No', width: 120, isPrimaryKey: true,
validationRules: { required: true } },
        { field: 'FIELD1', headerText: 'Player Name', width: 140, editType:
'dropdownedit', validationRules: { required: true } },
        { field: 'FIELD2', headerText: 'Year', width: 120, textAlign: 'Right' },
        { field: 'FIELD3', headerText: 'Stint', width: 120, textAlign: 'Right'
},
        { field: 'FIELD4', headerText: 'TMID', width: 120, textAlign: 'Right' },
        { field: 'FIELD5', headerText: 'LGID', width: 120, textAlign: 'Right' },
    ],

```

```

    { field: 'FIELD6', headerText: 'GP', width: 120, textAlign: 'Right' },
    { field: 'FIELD7', headerText: 'GS', width: 120, textAlign: 'Right' },
    { field: 'FIELD8', headerText: 'Minutes', width: 120, textAlign: 'Right'
  },
    { field: 'FIELD9', headerText: 'Points', width: 120, textAlign: 'Right'
  },
    { field: 'FIELD10', headerText: 'oRebounds', width: 130, textAlign:
'Reight' },
    { field: 'FIELD11', headerText: 'dRebounds', width: 130, textAlign:
'Reight' },
    { field: 'FIELD12', headerText: 'Rebounds', width: 120, textAlign:
'Reight' },
    { field: 'FIELD13', headerText: 'Assists', width: 120, textAlign:
'Reight' },
    { field: 'FIELD14', headerText: 'Steals', width: 120, textAlign: 'Right'
  },
    { field: 'FIELD15', headerText: 'Blocks', width: 120, textAlign: 'Right'
  },
    { field: 'FIELD16', headerText: 'Turnovers', width: 130, textAlign:
'Reight' },
    { field: 'FIELD17', headerText: 'PF', width: 130, textAlign: 'Right' },
    { field: 'FIELD18', headerText: 'fgAttempted', width: 150, textAlign:
'Reight' },
    { field: 'FIELD19', headerText: 'fgMade', width: 120, textAlign: 'Right'
  },
    { field: 'FIELD20', headerText: 'ftAttempted', width: 150, textAlign:
'Reight' },
    { field: 'FIELD21', headerText: 'ftMade', width: 120, textAlign: 'Right'
  },
    { field: 'FIELD22', headerText: 'ThreeAttempted', width: 150, textAlign:
'Reight' },
    { field: 'FIELD23', headerText: 'ThreeMade', width: 130, textAlign:
'Reight' },
    { field: 'FIELD24', headerText: 'PostGP', width: 120, textAlign: 'Right'
  },
    { field: 'FIELD25', headerText: 'PostGS', width: 120, textAlign: 'Right'
  },
    { field: 'FIELD26', headerText: 'PostMinutes', width: 120, textAlign:
'Reight' },
    { field: 'FIELD27', headerText: 'PostPoints', width: 130, textAlign:
'Reight' },
    { field: 'FIELD28', headerText: 'PostoRebounds', width: 130, textAlign:
'Reight' },
    { field: 'FIELD29', headerText: 'PostdRebounds', width: 130, textAlign:
'Reight' },
    { field: 'FIELD30', headerText: 'PostRebounds', width: 130, textAlign:
'Reight', editType: 'numericedit', validationRules: { required: true } }}
  });
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">

```

```

<meta name="description" content="Typescript Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>

```

```

<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Column's [width](#) is required for column virtualization. If column's width is not defined then Grid will consider its value as **200px**.

Virtualization with Grouping

Both the row and column virtualization can be used along with grouping. At initial rendering, the virtual height of scrollbar will be set based on the total number of records and after grouping, it will be refreshed based on the grouped state(expand/collapse). While collapse the group caption row in current viewport then the next view page grouped records are shown.

The collapsed/expanded state will persist only for local dataSource while scrolling.

Limitations for virtual scrolling

- While using column virtual scrolling, column width should be in the pixel. Percentage values are not accepted.
- Due to the element height limitation in browsers, the maximum number of records loaded by the grid is limited by the browser capability.
- The cell selection is not supported for both row and column virtual scrolling.
- Virtual scrolling is not compatible with batch editing, detail template, rowspan, colspan and hierarchy features.
- Group expand and collapse state will not be persisted.
- Since data is virtualized in grid, the aggregated information and total group items are displayed based on the current view items. To get these information regardless of the view items, refer to the

[Group with Page](#) topic.

- The page size provided must be two times larger than the number of visible rows in the grid. If the page size is failed to meet this condition then the size will be determined by grid.
- The height of the grid content is calculated using the row height and total number of records in the data source and hence features which changes row height such as text wrapping are not supported. If you want to increase the row height to accommodate the content then you can specify the row height as below to ensure all the table rows are in same height.

```

.e-grid .e-row {
height: 2em;
}

```

- Programmatic selection using the [selectRows](#) method is not supported in virtual scrolling.

Browser height limitation in virtual scrolling and solution

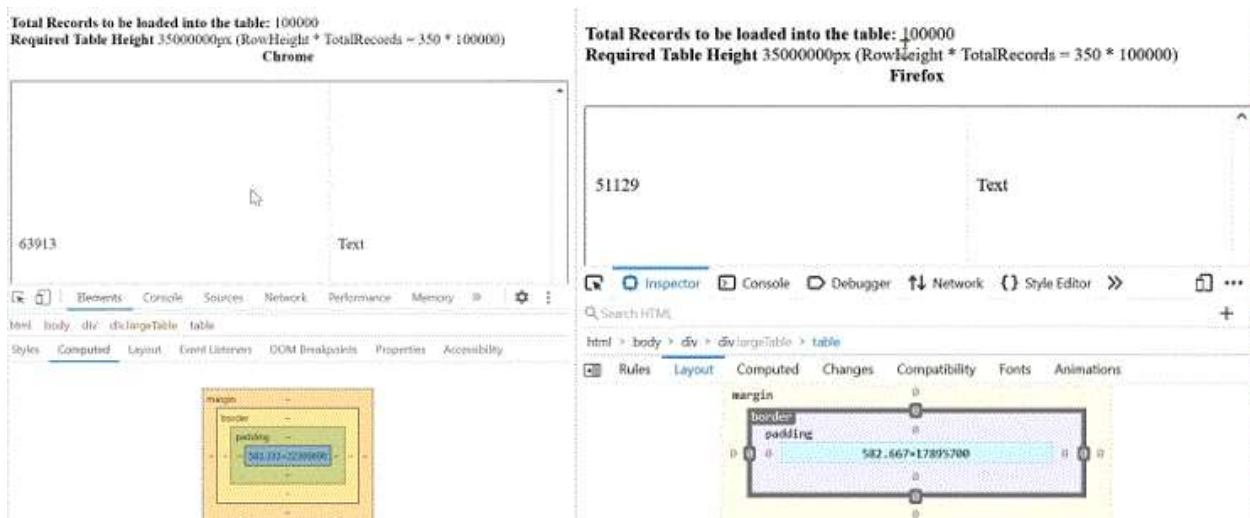
You can load millions of records in the Grid by using virtual scrolling, where the grid loads and renders rows on-demand while scrolling vertically. As a result, Grid lightens the browser's load by minimizing the DOM elements and rendering elements visible in the viewport. The height of the grid is calculated using the Total Records Count * [Row Height](#) property.

The browser has some maximum pixel height limitations for the scroll bar element. The content placed above the maximum height can't be scrolled if the element height is greater than the browser's maximum height limit. The browser height limit affects the virtual scrolling of the grid. When a large number of records are bound to the Grid, it can only display the records until the maximum height limit of the browser. Once the browser's height limit is reached while scrolling, the user won't be able to scroll further to view the remaining records.

For example, if the row height is set as 30px and the total record count is 1000000(1 million), then the height of the grid element will be 30,000,000 pixels. In this case, the browser's maximum height limit for a div is about 22,369,600 (The maximum pixel height limitation differs for different browsers). The records above the maximum height limit of the browser can't be scrolled.

This height limitation is not related to the Grid component. It fully depends on the default behavior of the browser. The same issue is reproduced in the normal HTML table too.

The following image illustrates the height limitation issue of a normal HTML table in different browsers (Chrome and Firefox).



Grid component also faced the same issue as mentioned in the below image.



The Grid has an option to overcome this limitation of the browser in the following ways.

Solution 1: Using external buttons

You can prevent the height limitation problem in the browser when scrolling through millions of records by loading the segment of data through different strategy.

In the following sample, Grid is rendered with a large number of records(nearly 2 million). Here, you can scroll 0.5 million records at a time in Grid. Once you reach the last page of 0.5 million records, the **Load Next Set** button will be shown at the bottom of the Grid. By clicking that button, you can view the next set of 0.5 million records in Grid. Also, the **Load Previous Set** button will be shown at the top of the Grid to load the previous set of 0.5 million records.

Let's see the step by step procedure for how we can overcome the limitation in the Syncfusion Grid component.

1. Create a custom adaptor by extending UrlAdaptor and binding it to the grid DataSource property. In the processQuery method of the custom adaptor, we handled the Skip query based on the current page set to perform the data operation with whole records on the server.

```
`ts
```

```
class CustomUrlAdaptor extends UrlAdaptor {
  processQuery(args) {
    if (arguments[1].queries) {
      for (var i = 0; i < arguments[1].queries.length; i++) {
        if (arguments[1].queries[i].fn === 'onPage') {
          // pageSet - defines the number of segments that we are going to split the 2million records. In this
          // example we have considered 0.5 million records for each set so the pageSet is 1, 2, 3 and 4.
          // maxRecordsPerPageSet – In this example we define the value as 0.5 million.
          // gridPageSize – the pageSize that we have defined in the Grid pageSettings->pageSize property
          // customize the pageIndex based on the current pageSet (It send the skip query including the previous
          // pageSet ) so that the other operations performed for total 2millions records instead of 0.5 million alone.
```

```

arguments[1].queries[i].e.pageIndex = (((pageSet - 1) * maxRecordsPerPageSet) / gridPageSize) +
arguments[1].queries[i].e.pageIndex;
}
}
}
let original = super.processQuery.apply(this, arguments);
return original;
}
}
let data: DataManager = new DataManager({
adaptor: new CustomUrlAdaptor,
url: "Home/UrlDatasource"
});
、

```

2. Render the grid by define the following features.

```

`ts
let grid: Grid = new Grid({
dataSource: data,
enableVirtualization: true,
pageSettings: {pageSize: 50},
height: 360,
beforeDataBound: beforeDataBound,
columns: [
{ field: 'OrderID', width: 120, headerText: 'Order ID', textAlign: 'Right' }
.....
.....
]
});
、

```

3. In the beforeDataBound event, we set the args.count as 0.5 million to perform scrolling with 0.5 million records and all the data operations are performed with whole records which is handled using the custom adaptor. And also particular segment records count is less than 0.5 million means it will directly assigned the original segmented count instead of 0.5 million.

```
`ts
beforeDataBound(args) {
// storing the total records count which means 2 million records count
totalRecords = args.count;
// change the count with respect to maxRecordsPerPageSet (maxRecordsPerPageSet = 500000)
args.count = args.count - ((pageSet - 1) maxRecordsPerPageSet) > maxRecordsPerPageSet
?maxRecordsPerPageSet : args.count - ((pageSet - 1) maxRecordsPerPageSet);
}
`
```

4. Render “Load Next Set” button and “Load Previous Set” button at bottom and top of the grid component.

```
`ts
let button: Button = new Button({
cssClass: 'e-info prevbtn',
onClick: prevBtnClick,
content: 'Load Previous Set...'
});
let grid: Grid = new Grid({
dataSource: data,
enableVirtualization: true,
pageSettings: {pageSize: 50},
height: 360,
beforeDataBound: beforeDataBound,
columns: [
{ field: 'OrderID', width: 120, headerText: 'Order ID', textAlign: 'Right' }
.....
.....
]
});
let button: Button = new Button({
cssClass: 'e-info nextbtn',
onClick: nextBtnClick,
content: 'Load Next Set...'
});
```



```
});  
`
```

5. While click on the **Load Next Set / Load Previous Set** button corresponding page data set is loaded to view remaining records of total 2 millions records after doing some simple calculation.

```
`ts
```

```
// Triggered when clicking the Previous/ Next button.
```

```
prevNxtBtnClick(args) {
```

```
if (grid.element.querySelector('.e-content') && grid.element.querySelector('.e-content').getAttribute('aria-busy') === 'false') {
```

```
// Increase/decrease the pageSet based on the target element.
```

```
pageSet = args.target.classList.contains('prevbtn') ? --pageSet : ++pageSet;
```

```
this.rerenderGrid(); // Re-render the Grid component.
```

```
}
```

```
}
```

```
`
```

You can view the hosted link for this sample [here](#).

LOAD PREVIOUS SET...				
Order ID	Customer ID	Freight	Country	Status
1	Alfki	\$12.30	📍Denmark	Active
2	Anatr	\$3.30	📍Brazil	Inactive
3	Anton	\$4.30	📍Germany	Active
4	Blomp	\$5.30	📍Austria	Inactive
5	Bolid	\$6.30	📍Switzerland	Active
6	Hanar	\$12.30	📍France	Active
7	Thomas	\$3.30	📍Itali	Inactive
8	Jack	\$4.30	📍Austria	Active
9	Alfki	\$5.30	📍USA	Inactive
10	Hanar	\$6.30	📍Belgium	Active
11	Alfki	\$3.30	📍Denmark	Active

| LOAD NEXT SET... | | | | |

If you perform grid actions such as filtering, sorting, etc., after scrolling through the 0.5 million data, the Grid performs those data actions with the whole records, not just the current loaded 0.5 million data.

Solution 2: Using RowHeight property

You can reduce the [row height](#) using the [rowHeight](#) property of the Grid. It will reduce the overall height to accommodate more rows. But this approach optimizes the limitation, but if the height limit is reached after reducing row height also, you have to opt for the previous solution or use paging.

In the following image, you can see how many records will be scrollable when setting rowHeight to "36px" and "30px".

The image displays two screenshots of a Grid control. The top screenshot shows a grid with 8 rows and 5 columns: Order ID, Customer ID, Weight, Country, and Status. The rows are numbered 1 to 8, and the status is either 'Active' or 'Inactive'. The bottom screenshot shows a similar grid but with 10 rows, including additional data for Order IDs 9 and 10. Both grids have a 'RowHeight' property set to 36px and 30px respectively.

Order ID	Customer ID	Weight	Country	Status
1	Alfo	\$12.50	Denmark	Active
2	Amir	\$3.30	Brazil	Inactive
3	Anton	\$4.50	Germany	Active
4	Bomp	\$5.30	Austria	Inactive
5	Bold	\$6.30	Switzerland	Active
6	Hansr	\$12.30	France	Active
7	Thomas	\$3.30	Italy	Inactive
8	Jack	\$4.30	Austria	Active
9	Alfo	\$8.30	USA	Inactive
10	Hansr	\$6.30	Belgium	Active

Solution 3: Using paging instead of virtual scrolling

Similar to virtual scrolling, the [paging](#) feature also loads the data in an on-demand concept. Pagination is also compatible with all the other features (Grouping, Editing, etc.) in Grid. So, use the paging feature instead of virtual scrolling to view a large number of records in the Grid without any kind of performance degradation or browser height limitation.

Infinite scroll in ##Platform_Name## Grid control

Infinite scrolling is used to load a huge amount of data without degrading the Grid performance. This feature works like the lazy loading concept, which means the buffer data is loaded only when the scrollbar reaches the end of the scroller.

To enable Infinite scrolling, set `enableInfiniteScrolling` property as true.

* In this feature, Grid will not make a new data request when you visit the same page again.

INDEX.JS

```
var names = ['TOM', 'Hawk', 'Jon', 'Chandler', 'Monica', 'Rachel', 'Phoebe', 'Gunter', 'Ross', 'Geller', 'Joey', 'Bing', 'Tribbiani', 'Janice', 'Bong', 'Perk', 'Green', 'Ken', 'Adams'];
var hours = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
```

```

var designation = ['Manager', 'Engineer 1', 'Engineer 2', 'Developer',
'Tester'];
var statusValue = ['Completed', 'Open', 'In Progress', 'Review', 'Testing']
var data = function(count) {
    var result = [];
    for (var i = 0; i < count; i++) {
        result.push({
            TaskID: i + 1,
            Engineer: names[Math.round(Math.random() * names.length)] ||
names[0],
            Designation: designation[Math.round(Math.random() *
designation.length)] || designation[0],
            Estimation: hours[Math.round(Math.random() * hours.length)] ||
hours[0],
            Status: statusValue[Math.round(Math.random() *
statusValue.length)] || statusValue[0]
        });
    }
    return result;
};
(window).getStatus = function(statusValue){
    var colors = { 'Completed': 'green', 'Open': 'red', 'In Progress':
'#FB1E77', 'Review': 'brown', 'Testing': '#1EC1FB' };
    return '<span style="color:' + colors[statusValue] + '">' + statusValue
+ '</span>';
};
ej.grids.Grid.Inject(ej.grids.InfiniteScroll);
var grid = new ej.grids.Grid({
    dataSource: data(1000),
    height: 300,
    enableInfiniteScrolling: true,
    pageSettings: { pageSize: 50 },
    columns: [
        { field: 'TaskID', headerText: 'Task ID', textAlign: 'Right', width:
50, type: 'number' },
        { field: 'Engineer', width: 100 },
        { field: 'Designation', width: 100 },
        { field: 'Estimation', textAlign: 'Right', width: 100 },
        { field: 'Status', width: 100, template: '${getStatus(data.Status)}'
    }
    ]
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}

```

```

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

InitialBlocks

You can define the initial loading pages count by using `infiniteScrollSettings.initialBlocks` property. By default, this feature loads three pages in initial rendering.

In the below demo, we have changed this property value to load five page records instead of three.

INDEX.JS

```

var names = ['TOM', 'Hawk', 'Jon', 'Chandler', 'Monica', 'Rachel', 'Phoebe',
'Gunter', 'Ross', 'Geller', 'Joey', 'Bing', 'Tribbiani', 'Janice', 'Bong',
'Perk', 'Green', 'Ken', 'Adams'];
var hours = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
var designation = ['Manager', 'Engineer 1', 'Engineer 2', 'Developer',
'Tester'];
var statusValue = ['Completed', 'Open', 'In Progress', 'Review', 'Testing']
var data = function(count) {
    var result = [];
    for (var i = 0; i < count; i++) {
        result.push({
            TaskID: i + 1,
            Engineer: names[Math.round(Math.random() * names.length)] ||
names[0],
            Designation: designation[Math.round(Math.random() *
designation.length)] || designation[0],
            Estimation: hours[Math.round(Math.random() * hours.length)] ||
hours[0],
            Status: statusValue[Math.round(Math.random() *
statusValue.length)] || statusValue[0]
        });
    }
    return result;
};
(window).getStatus = function(statusValue){
    var colors = { 'Completed': 'green', 'Open': 'red', 'In Progress':
'#FB1E77', 'Review': 'brown', 'Testing': '#1EC1FB' };
    return '<span style="color:' + colors[statusValue] + '>' + statusValue
+ '</span>';
};
ej.grids.Grid.Inject(ej.grids.InfiniteScroll);
var grid = new ej.grids.Grid({
    dataSource: data(1000),
    height: 300,
    enableInfiniteScrolling: true,
    infiniteScrollSettings: { initialBlocks: 5 },
    pageSettings: { pageSize: 50 },
    columns: [
        { field: 'TaskID', headerText: 'Task ID', textAlign: 'Right', width:
50, type: 'number' },
        { field: 'Engineer', width: 100 },
        { field: 'Designation', width: 100 },
        { field: 'Estimation', textAlign: 'Right', width: 100 },

```

```

        { field: 'Status', width: 100, template: '${getStatus(data.Status)}'
    }
    ]
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <style>
        .e-row[aria-selected="true"] .e-customizedExpandcell {
            background-color: #e0e0e0;
        }
        .e-grid.e-gridhover tr[role='row']:hover {
            background-color: #eee;
        }
        .e-expand::before {
            content: '\e5b8';
        }
        .empImage {
            margin: 6px 16px;
            float: left;
            width: 50px;
            height: 50px;
        }
    </style>

```

```

    }
    </style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Cache Mode

Cache is used to store the loaded rows object in the Grid instance which can be reused for creating the row elements whenever you scroll to already visited page. Also, this mode maintains row elements based on the `infiniteScrollSettings.maxBlocks` count value, once this limit exceeds then it will remove row elements from DOM for new rows.

To enable the cache mode in Infinite scrolling, set `infiniteScrollSettings.enableCache` property as true.

INDEX.JS

```

var names = ['TOM', 'Hawk', 'Jon', 'Chandler', 'Monica', 'Rachel', 'Phoebe',
'Gunter', 'Ross', 'Geller', 'Joey', 'Bing', 'Tribbiani', 'Janice', 'Bong',
'Perk', 'Green', 'Ken', 'Adams'];
var hours = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
var designation = ['Manager', 'Engineer 1', 'Engineer 2', 'Developer',
'Tester'];
var statusValue = ['Completed', 'Open', 'In Progress', 'Review', 'Testing']
var data = function(count) {
    var result = [];
    for (var i = 0; i < count; i++) {
        result.push({
            TaskID: i + 1,
            Engineer: names[Math.round(Math.random() * names.length)] ||
names[0],
            Designation: designation[Math.round(Math.random() *
designation.length)] || designation[0],
            Estimation: hours[Math.round(Math.random() * hours.length)] ||
hours[0],
            Status: statusValue[Math.round(Math.random() *
statusValue.length)] || statusValue[0]
        });
    }
    return result;
};

```



```

(window).getStatus = function(statusValue){
    var colors = { 'Completed': 'green', 'Open': 'red', 'In Progress':
    '#FB1E77', 'Review': 'brown', 'Testing': '#1EC1FB' };
    return '<span style="color:' + colors[statusValue] + '">' + statusValue
+ '</span>';
};
ej.grids.Grid.Inject(ej.grids.InfiniteScroll);
var grid = new ej.grids.Grid({
    dataSource: data(1000),
    height: 300,
    enableInfiniteScrolling: true,
    infiniteScrollSettings: { enableCache: true },
    pageSettings: { pageSize: 50 },
    columns: [
        { field: 'TaskID', headerText: 'Task ID', textAlign: 'Right', width:
50, type: 'number' },
        { field: 'Engineer', width: 100 },
        { field: 'Designation', width: 100 },
        { field: 'Estimation', textAlign: 'Right', width: 100 },
        { field: 'Status', width: 100, template: '${getStatus(data.Status)}'
    }
    ]
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Limitations for Infinite Scrolling

- Due to the element height limitation in browsers, the maximum number of records loaded by the grid is limited due to the browser capability.
- Initial loading rows total height must be greater than the viewport height.
- Cell selection will not be persisted in cache mode.
- Infinite scrolling is not compatible with batch editing, detail template and hierarchy features.
- The group records cannot be collapsed in cache mode.
- The aggregated information and total group items are displayed based on the current view items. To get these information regardless of the view items, refer to the

[Group with Page](#) topic.

- Programmatic selection using the [selectRows](#) and [selectRow](#) method is not supported in infinite scrolling.

Selection

Selection in ##Platform_Name## Grid control

Selection provides an option to highlight a row or a cell or a column. It can be done through simple mouse down or arrow keys. To disable selection in the Grid, set the [allowSelection](#) to false.

The grid supports two types of selection that can be set by using the [selectionSettings.type](#). They are:

- **Single:** The **Single** value is set by default, and it only allows selection of a single row or a cell or a column.
- **Multiple:** Allows you to select multiple rows or cells or columns.

To perform the multi-selection, press and hold CTRL key and click the desired rows or cells or columns. To select range of rows or cells or columns, press and hold the SHIFT key and click the rows or cells or columns.

INDEX.JS

```
var grid = new ej.grids.Grid({
  dataSource: data,
  selectionSettings: { type: 'Multiple' },
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
    { field: 'ShipCity', headerText: 'Ship City', width: 150 },
    { field: 'ShipName', headerText: 'Ship Name', width: 150 }
  ],
  height: 315
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Selection mode

The grid supports three types of selection mode that can be set by using

the [selectionSettings.mode](#). They are:

- **Row:** The **Row** value is set by default, and allows you to select only rows.
- **Cell:** Allows you to select only cells.
- **Both:** Allows you to select rows and cells at the same time.

INDEX.JS

```
var grid = new ej.grids.Grid({
  dataSource: data,
  selectionSettings: { mode: 'Both' },
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
    { field: 'ShipCity', headerText: 'Ship City', width: 150 },
    { field: 'ShipName', headerText: 'Ship Name', width: 150 }
  ],
  height: 315
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
```

```

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Touch interaction

When you tap a grid row on touchscreen device, the tapped row is selected. It also shows a popup

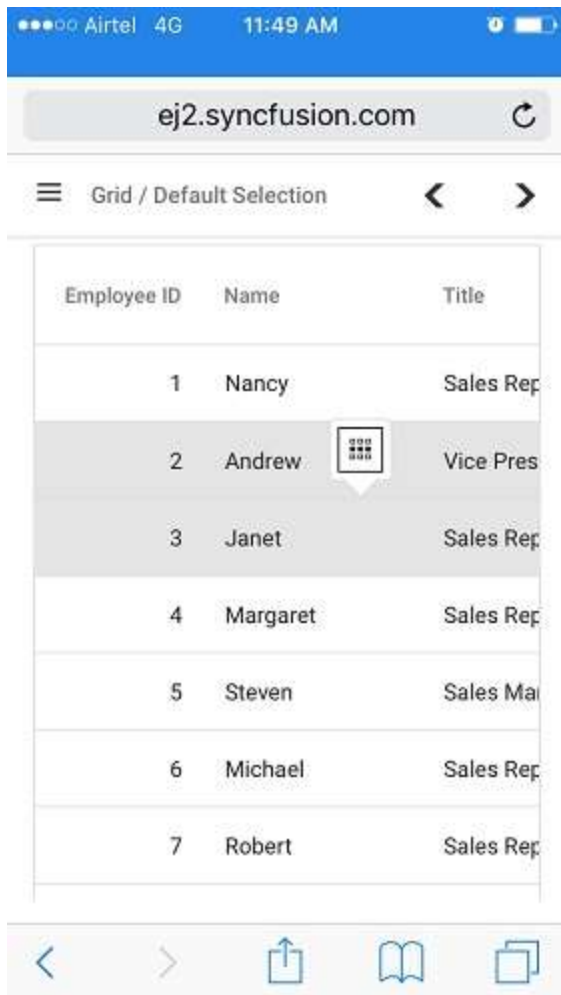


for multi-row selection. To select multiple rows or cells, tap the popup and then tap the desired rows or cells.



Multi-selection requires the selection [type](#) to be **multiple**.

The following screenshot represents a grid touch selection in the device.



Row selection in `##Platform_Name##` Grid control

Select row at initial rendering

To select a row at initial rendering, set the [selectedRowIndex](#) value.

INDEX.JS

```
var grid= new ej.grids.Grid({
  dataSource: data,
  selectionSettings: { type: 'Multiple', mode: 'Both' },
  selectedRowIndex: 1,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
    { field: 'ShipCity', headerText: 'Ship City', width: 150 },
    { field: 'ShipName', headerText: 'Ship Name', width: 150 }
  ],
  height: 315
});
grid.appendTo('#Grid');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
      background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
      background-color: #eee;
    }
    .e-expand::before {
      content: '\e5b8';
    }
    .empImage {
      margin: 6px 16px;
      float: left;
      width: 50px;
      height: 50px;
    }
  </style>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```



```

    <div id="container">
        <div id="Grid"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Get selected row indexes

You can get the selected row indexes by using the [getSelectedRowIndexes](#) method.

INDEX.JS

```

var grid = new ej.grids.Grid({
    dataSource: data,
    selectionSettings: {type: 'Multiple'},
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'ShipCity', headerText: 'Ship City', width: 150 },
        { field: 'ShipName', headerText: 'Ship Name', width: 150 }
    ],
    height: 315,
    rowSelected: rowSelected
});
grid.appendTo('#Grid');
function rowSelected(args) {
    var selectedrowindex = grid.getSelectedRowIndexes(); // get the
selected row indexes.
    alert(selectedrowindex); // to alert the selected row indexes.
    var selectedrecords = grid.getSelectedRecords(); // get the selected
records.
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Simple multiple row selection

You can select multiple rows by clicking on rows one by one. This will not deselect the previously selected rows. To deselect the previously selected row, you can click on the selected row. You can enable this behavior by using [selectionSettings.enableSimpleMultiRowSelection](#) property.

INDEX.JS

```

var grid = new ej.grids.Grid({
    dataSource: sdata,
    allowSelection: true,
    selectionSettings: {type: 'Multiple', enableSimpleMultiRowSelection:
true},
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },

```

```

        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'ShipCountry', headerText: 'Ship Country', width: 150 },
        { field: 'ShipName', headerText: 'Ship Name', width: 150 }
    ]
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');

```

```

if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Toggle selection

The Toggle selection allows to perform selection and unselection of the particular row or cell or column. To enable toggle selection, set [enableToggle](#) property of the selectionSettings as true. If you click on the selected row or cell or column then it will be unselected and vice versa.

INDEX.JS

```

var grid = new ej.grids.Grid({
    dataSource: data,
    selectionSettings: {enableToggle: true},
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'ShipCity', headerText: 'Ship City', width: 150 },
        { field: 'ShipName', headerText: 'Ship Name', width: 150 }
    ],
    height: 315
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

* If multi selection is enabled, then first click on any selected row (without pressing Ctrl key), it will clear the multi selection and in second click on the same row, it will be unselected.

Clear selection programmatically

You can clear the Grid selection programmatically by using the [clearSelection](#) method.

In the demo below, we initially selected the third row using [selectedRowIndex](#). You can clear this selection by calling the [clearSelection](#) method in the button click event.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Page, ej.grids.Selection);
var grid = new ej.grids.Grid({
  dataSource: data,
  allowSelection: true,
  allowPaging: true,
  selectionSettings: { type: 'Multiple' },
  selectedRowIndex: 2,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
    { field: 'ShipCountry', headerText: 'Ship Country', width: 130 },
    { field: 'Freight', headerText: 'Freight', format: 'C2', width: 100
}
  ],
  pageSettings: { pageSizes: true, pageSize: 5 }
});
grid.appendTo('#Grid');
var btn = new ej.buttons.Button({ cssClass: 'e-flat' }, '#show');
document.getElementById('show').onclick = () => {
  grid.clearSelection();
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="show">Clear Selection</button>
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Get selected records on various pages

Enabling the [selectionSettings.persistSelection](#) property will persist the selection in all Grid operations.

So the selection will be maintained on every page even after navigating to another page.

You can get the selected records using the [getSelectedRecords](#) method.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Page, ej.grids.Selection);
var grid = new ej.grids.Grid({
    dataSource: data,
    allowSelection: true,
    allowPaging: true,
    selectionSettings: { type: 'Multiple', persistSelection: true },
    columns: [
        { type: 'checkbox', width: 50 },
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
isPrimaryKey: true, width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
        { field: 'ShipCountry', headerText: 'Ship Country', width: 130 },
        { field: 'Freight', headerText: 'Freight', format: 'C2', width: 100
    }
    ],
    pageSettings: { pageSizes: true, pageSize: 5 }
});
grid.appendTo('#Grid');
var btn = new ej.buttons.Button({ cssClass: 'e-flat' }, '#show');
document.getElementById('show').onclick = () => {
    var selectedrecords = grid.getSelectedRecords();
    var selectedRecordsCount = selectedrecords.length;
    alert(selectedRecordsCount);
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
      background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
      background-color: #eee;
    }
    .e-expand::before {
      content: '\e5b8';
    }
  </style>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```



```

<script id="template" type="text/x-template">
  <input id='${OrderID}' value='${Freight}' class='custemp'
type='text' style='width: 100%'>
</script>
<div id="container">
  <button id="show">Selected Records</button>
  <div id="Grid"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

* To persist the grid selection, it is necessary to define any one of the columns as a primary key using the [columns.isPrimaryKey](#) property.

Cell selection in ##Platform_Name## Grid control

Cell selection can be done through simple mouse down or arrow keys (up, down, left, and right).

The grid supports two types of cell selection mode that can be set by using the [selectionSettings.cellSelectionMode](#). They are:

- **Flow:** The **Flow** value is set by default. The range of cells are selected between the start index and end index that includes in between cells of rows.
- **Box:** Range of cells are selected from the start and end column indexes that includes in between cells of rows within the range.

INDEX.JS

```

var grid = new ej.grids.Grid({
  dataSource: data,
  selectionSettings: { cellSelectionMode: 'Box', type: 'Multiple', mode:
'Cell' },
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
    { field: 'ShipCity', headerText: 'Ship City', width: 150 },
    { field: 'ShipName', headerText: 'Ship Name', width: 150 }
  ],
  height: 315
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">

```

```

<meta name="description" content="Typescript Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>

```

```
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Cell selection requires the [selectionSettings.mode](#) to be **Cell** or **Both**, and [type](#) should be **Multiple**.

Toggle selection

The Toggle selection allows to perform selection and unselection of the particular row or cell or column. To enable toggle selection, set [enableToggle](#) property of the selectionSettings as true. If you click on the selected row or cell or column then it will be unselected and vice versa.

INDEX.JS

```
var grid = new ej.grids.Grid({
    dataSource: data,
    selectionSettings: {enableToggle: true},
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'ShipCity', headerText: 'Ship City', width: 150 },
        { field: 'ShipName', headerText: 'Ship Name', width: 150 }
    ],
    height: 315
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

* If multi selection is enabled, then first click on any selected row (without pressing Ctrl key), it will clear the multi selection and in second click on the same row, it will be unselected.

Column selection in ##Platform_Name## Grid control

Column selection can be done through simple mouse down or arrow keys.

You can enable column selection by setting the [selectionSettings.allowColumnSelection](#) property as true.

INDEX.JS

```
var grid = new ej.grids.Grid({
  dataSource: data,
  selectionSettings: { allowColumnSelection: true, type: 'Multiple' },
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
    { field: 'ShipCity', headerText: 'Ship City', width: 150 },
    { field: 'ShipName', headerText: 'Ship Name', width: 150 }
  ],
  height: 315
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Check box selection in ##Platform_Name## Grid control

Checkbox selection provides an option to select multiple grid records with help of checkbox in each row.

To render the checkbox in each grid row, you need to use checkbox column with type as `checkbox` using the column `type` property.

INDEX.JS

```

var grid = new ej.grids.Grid({
    dataSource: data,
    columns: [
        { type: 'checkbox' },
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'ShipCity', headerText: 'Ship City', width: 150 },
        { field: 'ShipName', headerText: 'Ship Name', width: 150 }
    ],
    height: 315
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>

```

```
</body></html>
```

* By default, selection is allowed by clicking a grid row or checkbox in that row. To allow selection only through checkbox, you can set the

[selectionSettings.checkboxOnly](#) property to true.

* Selection can be persisted in all the operations using the [selectionSettings.persistSelection](#) property. For persisting selection on the grid, any one of the columns should be defined as a primary key using the [columns.isPrimaryKey](#) property.

Checkbox selection mode

In checkbox selection, selection can also be done by clicking on rows. This selection provides two types of Checkbox Selection mode which can be set by using the following API,

[selectionSettings.checkboxMode](#). The modes are;

- **Default:** This is the default value of the checkboxMode. In this mode, user can select multiple rows by clicking rows one by one.
- **ResetOnRowClick:** In ResetOnRowClick mode, when user clicks on a row it will reset previously selected row. Also you can perform multiple-selection in this mode by press

and hold CTRL key and click the desired rows. To select range of rows, press and hold the SHIFT key and click the rows.

INDEX.JS

```
var grid = new ej.grids.Grid({
  dataSource: data,
  selectionSettings: {checkboxMode: 'ResetOnRowClick'},
  columns: [
    { type: 'checkbox', width: 50 },
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
    { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
    { field: 'ShipCountry', width: 140, headerText: 'Ship Country',
visible: false },
    { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C' },
    { field: 'OrderDate', headerText: 'Order Date', width: 140, format:
'yMd', textAlign: 'Right' }
  ],
  height: 315
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
```



```

<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');

```

```

if(ele) {
    ele.style.visibility = "visible";
}

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Prevent specific rows from being selected in checkbox selection

You can prevent specific rows from being selected in the checkbox selection mode by hiding the checkboxes using the [rowDataBound](#) event. You achieve this by setting the [isSelectable](#) argument as false in the [rowDataBound](#) event based on certain conditions as per the needs of the application.

In the following sample, the selection of specific rows has been prevented based on the [isSelectable](#) argument in the [rowDataBound](#) event.

INDEX.JS

```

var grid = new ej.grids.Grid({
    dataSource: data,
    selectionSettings: { persistSelection: true },
    allowFiltering: true,
    filterSettings: { type: 'CheckBox' },
    pageSettings: { pageSize: 20 },
    editSettings: {
        allowEditing: true,
        allowAdding: true,
        allowDeleting: true,
        mode: 'Normal',
    },
    toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel', 'Search'],
    height: 400,
    columns: [
        { type: 'checkbox', width: 120 },
        { field: 'List', headerText: 'List', width: 120 },
        { field: 'OrderID', isPrimaryKey: true, headerText: 'Order ID',
width: 150 },
        { field: 'CustomerID', headerText: 'CustomerID', width: 150 },
        { field: 'EmployeeID', headerText: 'Employee ID', width: 150 },
        { field: 'ShipCity', headerText: 'Ship City', width: 150 }
    ]
});
grid.appendTo('#Grid');
grid.rowDataBound = function(args) {
    args.isSelectable = args.data.List % 5 === 0;
}
for (var i = 0; i < data.length; i++) {
    data[i]['List'] = i + 1;
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

```

```

<meta name="description" content="Typescript Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>

```

```
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Aggregates

Aggregates in ##Platform_Name## Grid control

Aggregate values are displayed in the footer, group footer, or group caption of the Grid. It can be configured through `aggregates` property.

[Field](#) and [type](#) are the minimum properties required to represent an aggregate column.

To use the aggregate feature, you have to inject the `Aggregate` module.

By default, the aggregate value can be displayed in the footer, group, and caption cells. To show the aggregate value in one of the cells, use the [footerTemplate](#), [groupFooterTemplate](#), or [groupCaptionTemplate](#) property.

Built-in aggregate types

The aggregate type should be specified in the [type](#) property to configure an aggregate column.

The built-in aggregates are,

- Sum
- Average
- Min
- Max
- Count
- Truecount
- Falsecount

* Multiple aggregates can be used for an aggregate column by setting the [type](#) property with an array of aggregate types.

* Multiple types for a column is supported only when one of the aggregate templates is used.

Footer aggregate in ##Platform_Name## Grid control

Footer aggregate value is calculated for all the rows, and it is displayed in the footer cells. Use the [footerTemplate](#) property to render the aggregate value in footer cells.

INDEX.JS

```
ej.grids.Grid.Inject(ej.grids.Aggregate);
var grid = new ej.grids.Grid({
    dataSource: data,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
```

```

    { field: 'OrderDate', headerText: 'Order Date', width: 120, format:
'yMd' },
    { field: 'Freight', headerText: 'Freight', width: 150, format: 'C2'
},
    { field: 'ShipCountry', headerText: 'Ship Country', width: 150 }
],
height: 290,
aggregates: [{
    columns: [{
        type: 'Sum',
        field: 'Freight',
        format: 'C2',
        footerTemplate: 'Sum: ${Sum}'
    }]
},
{
    columns: [{
        type: 'Max',
        field: 'Freight',
        format: 'C2',
        footerTemplate: 'Max: ${Max}'
    }]
}]
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
        <div id="toolbar-template">
            <div id="refresh" title="Refresh">
                <button class="e-btn e-flat">
                    <span class="e-btn-icon e-icons e-refresh"></span>
                </button>
            </div>
        </div>
    </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The aggregate values must be accessed inside the template using their corresponding [type](#) name.

How to format aggregate value

You can format the aggregate value result by using the [format](#) property.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Aggregate);
var grid = new ej.grids.Grid({
    dataSource: data,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'OrderDate', headerText: 'Order Date', width: 120, format:
'yMd' },
        { field: 'Freight', headerText: 'Freight', width: 150, format: 'C2'
},
        { field: 'ShipCountry', headerText: 'Ship Country', width: 150 }
    ],
    height: 290,
    aggregates: [{
        columns: [{

```

```

        type: 'Sum',
        field: 'Freight',
        format: 'C2',
        footerTemplate: 'Sum: ${Sum}'
    ]
},
{
    columns: [{
        type: 'Max',
        field: 'Freight',
        format: 'C2',
        footerTemplate: 'Max: ${Max}'
    }]
}]
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>

```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
        <div id="toolbar-template">
            <div id="refresh" title="Refresh">
                <button class="e-btn e-flat">
                    <span class="e-btn-icon e-icons e-refresh"></span>
                </button>
            </div>
        </div>
    </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

How to place aggregates on top of the Grid

By default, the aggregated values are placed at the bottom of the footer section. It is possible to place the aggregated values at the top of the header. This is achieved by using the [dataBound](#) event, [getHeaderContent](#), and [getFooterContent](#) methods of the Grid.

In the following sample, the footer element is appended to the header element using the [getHeaderContent](#) and [getFooterContent](#) methods in the [dataBound](#) event.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Aggregate);
var grid = new ej.grids.Grid({
    dataSource: data,
    dataBound: dataBound,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'OrderDate', headerText: 'Order Date', width: 120, format:
'yMd' },
        { field: 'Freight', headerText: 'Freight', width: 150, format: 'C2'
},
        { field: 'ShipCountry', headerText: 'Ship Country', width: 150 }
    ],
    height: 290,
    aggregates: [{
        columns: [{
            type: 'Sum',
            field: 'Freight',
            format: 'C2',
            footerTemplate: 'Sum: ${Sum}'
        }]
    }]

```



```

    },
    {
        columns: [{
            type: 'Max',
            field: 'Freight',
            format: 'C2',
            footerTemplate: 'Max: ${Max}'
        }]
    }]
});
grid.appendTo('#Grid');
function dataBound() {
    grid.getHeaderContent().append(grid.getFooterContent());
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>

```

```

</head>
<body>

    <div id="container">
        <div id="Grid"></div>
        <div id="toolbar-template">
            <div id="refresh" title="Refresh">
                <button class="e-btn e-flat">
                    <span class="e-btn-icon e-icons e-refresh"></span>
                </button>
            </div>
        </div>
    </div>
</body>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Group and caption aggregate in ##Platform_Name## Grid control

Group and caption aggregate values are calculated from the current group items. If [groupFooterTemplate](#) is provided, the aggregate values are displayed in the group footer cells; and if [groupCaptionTemplate](#) is provided, aggregate values are displayed in the group caption cells.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Group, ej.grids.Aggregate);
var grid = new ej.grids.Grid({
    dataSource: data,
    allowGrouping: true,
    groupSettings: { showDropArea: false, columns: ['ShipCountry'] },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'OrderDate', headerText: 'Order Date', width: 120, format:
'yMd' },
        { field: 'Freight', headerText: 'Freight', width: 150, format: 'C2'
},
        { field: 'ShipCountry', headerText: 'Ship Country', width: 150 }
    ],
    height: 290,
    aggregates: [{
        columns: [{
            type: 'Sum',
            field: 'Freight',
            format: 'C2',
            groupFooterTemplate: 'Sum: ${Sum}'
        }]
    },
    {
        columns: [{
            type: 'Max',

```

```

        field: 'Freight',
        format: 'C2',
        groupCaptionTemplate: 'Max: ${Max}'
    }
  ]
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Grid"></div>
    <div id="toolbar-template">
      <div id="refresh" title="Refresh">

```

```

        <button class="e-btn e-flat">
            <span class="e-btn-icon e-icons e-refresh"></span>
        </button>
    </div>
</div>
</div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The aggregate values must be accessed inside the template using their corresponding [type](#) name.

Custom aggregate in ##Platform_Name## Grid control

To calculate the aggregate value with your own aggregate functions, use the custom aggregate option. To use custom aggregation, specify the [type](#) as **Custom**, and provide the custom aggregate function in the [customAggregate](#) property.

The custom aggregate function will be invoked with different arguments for total and group aggregations.

- **Total aggregation:** The custom aggregate function will be called with the whole data and current [AggregateColumn](#)

object.

- **Group aggregation:** This will be called with the current group details and [AggregateColumn](#) object.

INDEX.JS

```

var customAggregateFn = function(data, aggColumn){
    return data.result.filter(function(item) {
        return item[aggColumn.columnName] === 'Brazil';
    }).length;
}
var grid = new ej.grids.Grid({
    dataSource: data,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'Freight', headerText: 'Freight', width: 150, format: 'C2'
},
        { field: 'ShipCountry', headerText: 'Ship Name', width: 150 }
    ],
    height: 268,
    aggregates: [{

```

```

        columns: [{
            type: 'Custom',
            customAggregate: customAggregateFn,
            columnName: 'ShipCountry',
            footerTemplate: 'Brazil Count: ${Custom}'
        }]
    });
    grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Grid"></div>

```

```

        <div id="toolbar-template">
            <div id="refresh" title="Refresh">
                <button class="e-btn e-flat">
                    <span class="e-btn-icon e-icons e-refresh"></span>
                </button>
            </div>
        </div>
    </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

To access the custom aggregate value inside the template, use the key as **Custom**.

Show the count of distinct values in aggregate row

You can calculate the aggregate value with your own aggregate functions. To use custom aggregation, specify the **type** as **Custom**, and provide the custom aggregate function in the [customAggregate](#) property.

In this below demo, we have show the count of distinct value for **ShipCountry** column by using custom aggregate.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Page, ej.grids.Aggregate);
var customAggregateFn = function() {
    let results = new ej.data.DataManager(this.currentViewData).executeLocal(new
    ej.data.Query().select(['ShipCountry']));
    let distinct = new ej.data.DataUtil.distinct(results, 'ShipCountry', true);
    return distinct.length;
}
var grid = new ej.grids.Grid({
    dataSource: data,
    allowPaging: true,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
        width: 120, type: 'number' },
        { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
        'string' },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
        width: 120, format: 'C2' },
        { field: "ShipCountry", headerText: "Ship Country", width: 150 }
    ],
    height: 220,
    aggregates: [{
        columns: [{
            type: 'Custom',
            customAggregate: customAggregateFn,
            columnName: 'ShipCountry',
            footerTemplate: 'Distinct Count: ${Custom}'
        }]
    }]
}

```

```

    ]]
  });
  grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Grid"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}

```

```

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Reactive aggregate in ##Platform_Name## Grid control

Auto update aggregate value in batch editing

When using batch editing, the aggregate values will be refreshed on every cell save. The footer, group footer, and group caption aggregate values will be refreshed.

Adding a new record to the grouped grid will not refresh the aggregate values.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Page, ej.grids.Toolbar, ej.grids.Edit,
ej.grids.Group, ej.grids.Aggregate);
var grid = new ej.grids.Grid({
  dataSource: data,
  allowPaging: true,
  pageSettings: { pageSize: 6 },
  toolbar: ['Delete', 'Update', 'Cancel'],
  editSettings: { allowEditing: true, allowDeleting: true, mode: 'Batch'
},
  allowGrouping: true,
  groupSettings: { showDropArea: false, columns: ['ShipCountry'] },
  columns: [
    { field: 'OrderID', headerText: 'Order ID', isPrimaryKey: true,
textAlign: 'Right', width: 120 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
    { field: 'Freight', headerText: 'Freight', width: 150, format: 'C2'
},
    { field: 'ShipCountry', headerText: 'Ship Name', width: 150 }
  ],
  height: 268,
  aggregates: [{
    columns: [
      {
        type: 'Sum',
        field: 'Freight',
        format: 'C2',
        footerTemplate: 'Sum : ${Sum}'
      }
    ]
  },
  {
    columns: [{
      type: 'Sum',
      field: 'Freight',
      format: 'C2',
      groupCaptionTemplate: 'Sum : ${Sum}'
    }
  ]
},
  {
    columns: [{
      type: 'Sum',
      field: 'Freight',
      format: 'C2',
      groupFooterTemplate: 'Sum : ${Sum}'
    }
  ]
}

```



```

    ]]
  }
]
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Grid"></div>
    <div id="toolbar-template">
      <div id="refresh" title="Refresh">
        <button class="e-btn e-flat">
          <span class="e-btn-icon e-icons e-refresh"></span>

```

```

        </button>
    </div>
</div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Refresh aggregate values in inline editing

By default, reactive aggregate update is not supported by inline and dialog edit modes as it is not feasible to anticipate the value change event for every editor. But, you can refresh the aggregates manually in the inline edit mode using the refresh method of aggregate module.

In the following code, the input event for the Freight column editor has been registered and the aggregate value has been refreshed manually.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Page, ej.grids.Toolbar, ej.grids.Edit,
ej.grids.Aggregate);
var grid = new ej.grids.Grid({
    dataSource: data,
    allowPaging:true,
    pageSettings:{pageSize:7},
    toolbar: ['Edit', 'Delete', 'Update', 'Cancel'],
    editSettings: { allowEditing: true, allowDeleting: true, mode: 'Normal'
},
    actionBegin: function(args){
        if(args.requestType === 'beginEdit'){
            selectedRecord = {};
            selectedRecord = args.rowData;
        };
    },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', isPrimaryKey:true,
textAlign: 'Right', width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'Freight', headerText: 'Freight', editType: 'numericedit',
format: 'C2', edit: { params: { change: funChange } },
width: 150},
        { field: 'ShipCountry', headerText: 'Ship Name', width: 150 }
    ],
    height: 268,
    aggregates: [{
        columns:[{
            type:'Sum',
            field:'Freight',
            format:'C2',
            footerTemplate:'Sum : ${Sum}'
        }]
    }]
}]

```

```

});
grid.appendTo('#Grid');
function funChange(args) {
    let gridObj = document.getElementById('Grid')['ej2_instances'][0];
    selectedRecords['Freight'] = args.value; // Set the edited value to
    aggregate column
    gridObj.aggregateModule.refresh(selectedRecords) // Refresh aggregates
    using edited data
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>

```

```

        <div id="toolbar-template">
            <div id="refresh" title="Refresh">
                <button class="e-btn e-flat">
                    <span class="e-btn-icon e-icons e-refresh"></span>
                </button>
            </div>
        </div>
    </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Print in ##Platform_Name## Grid control

To print the Grid, use the [print](#) method from grid instance. The print option can be displayed on the [toolbar](#) by adding the `print` toolbar item.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Toolbar);
var grid = new ej.grids.Grid({
    dataSource: data,
    toolbar: ['Print'],
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'ShipCity', headerText: 'Ship City', width: 150 },
        { field: 'ShipName', headerText: 'Ship Name', width: 150 }
    ],
    height: 272
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Page setup

Some of the print options cannot be configured through JavaScript code. So, you have to customize the layout, paper size, and margin options using the browser page setup dialog. Please refer to the following links to know more about the browser page setup:

- [Chrome](#)
- [Firefox](#)
- [Safari](#)
- [IE](#)

Print using an external button

To print the grid from an external button, invoke the [print](#) method.

INDEX.JS

```
var grid = new ej.grids.Grid({
  dataSource: data,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
    { field: 'ShipCity', headerText: 'Ship City', width: 150 },
    { field: 'ShipName', headerText: 'Ship Name', width: 150 }
  ],
  height: 280
});
grid.appendTo('#Grid');
var printBtn = new ej.buttons.Button();
printBtn.appendTo('#print');
document.getElementById('print').addEventListener('click', function(){
  grid.print();
});
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="print">Print</button>
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Print the visible page

By default, the grid prints all the pages. To print the current page alone, set the [printMode](#) to **CurrentPage**.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Toolbar);
var grid = new ej.grids.Grid({
    dataSource: data,
    toolbar: ['Print'],
    printMode: 'CurrentPage',
    allowPaging: true,
    pageSettings: { pageSize: 6 },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'ShipCity', headerText: 'Ship City', width: 150 },
        { field: 'ShipName', headerText: 'Ship Name', width: 150 }
    ]
}

```

```
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
      background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
      background-color: #eee;
    }
    .e-expand::before {
      content: '\e5b8';
    }
    .empImage {
      margin: 6px 16px;
      float: left;
      width: 50px;
      height: 50px;
    }
  </style>
```



```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Print the hierarchy grid

By default, the grid will be print the master and expanded child grids alone. you can change the print option by using the [hierarchyPrintMode](#) property. The available options are,

Mode	Behavior
----- -----	
Expanded	Prints the master grid with expanded child grids.
All	Prints the master grid with all the child grids.
None	Prints the master grid alone.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.DetailRow, ej.grids.Toolbar);
var grid= new ej.grids.Grid({
    dataSource: employeeData,
    toolbar: ["Print"],
    hierarchyPrintMode: 'All',
    columns: [
        { field: 'EmployeeID', headerText: 'Employee ID', textAlign:
'Right', width: 120 },
        { field: 'FirstName', headerText: 'First Name', width: 150 },
        { field: 'City', headerText: 'City', width: 150 },
        { field: 'Country', headerText: 'Country', width: 150 }
    ],
    childGrid: {
        dataSource: data,
        queryString: 'EmployeeID',
        columns: [
            { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
            { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
            { field: 'ShipCity', headerText: 'Ship City', width: 150 },
            { field: 'ShipName', headerText: 'Ship Name', width: 150 }
        ]
    }
});

```

```

    ],
  }
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Grid"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";

```

```

}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Print the master detail grid

The Grid has the option to visualize details of a record in another Grid in a master and detailed manner. By default, Grid will print the master grid alone. Instead of this, it is possible to print both the master and detail grids by using the [beforePrint](#) event of the Grid.

In the following sample, the detail grid is added to the `element` argument of the `beforePrint` event, resulting in both the master and detail grids being printed on the page.

INDEX.JS

```

var names = ['AROUT', 'BERGS', 'BLONP', 'CHOPS', 'ERNSH'];
var masterdata = customerData.filter(function(e) {
    return names.indexOf(e.CustomerID) !== -1;
});
var mastergrid = new ej.grids.Grid({
    dataSource: masterdata,
    toolbar: ['Print'],
    selectedRowIndex: 1,
    beforePrint: beforePrint,
    columns: [
        { field: 'ContactName', headerText: 'Customer Name', width: 150 },
        { field: 'CompanyName', headerText: 'Company Name', width: 150 },
        { field: 'Address', headerText: 'Address', width: 150 },
        { field: 'Country', headerText: 'Country', width: 130 },
    ],
    rowSelected: rowSelected,
});
mastergrid.appendTo('#MasterGrid');
function rowSelected(args) {
    let selectedRecord = args.data;
    grid.dataSource = data.filter((record) => record.CustomerName ===
selectedRecord.ContactName).slice(0, 5);
    document.getElementById('key').innerHTML = selectedRecord.ContactName;
}
function beforePrint(args) {
    let customEle = document.createElement('div');
    customEle.innerHTML = document.getElementsByClassName('e-
statustext')[0].innerHTML + grid.element.innerHTML;
    customEle.appendChild(document.createElement('br'));
    args.element.append(customEle);
}
var grid = new ej.grids.Grid({
    allowSelection: false,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', width: 100, textAlign:
'Right' },
        { field: 'Freight', headerText: 'Freight', width: 100, format: 'C2',
type: 'number' },
        { field: 'ShipName', headerText: 'Ship Name', width: 200 },
        { field: 'ShipCountry', headerText: 'Ship Country', width: 150 },
        { field: 'ShipAddress', headerText: 'Ship Address', width: 200 },
    ],
});

```

```

    ],
  });
  grid.appendTo('#DetailGrid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <p class="e-mastertext">Master Grid</p>
    <div id="MasterGrid">
    </div>
    <p></p><div class="e-statustext"> Showing orders of Customer:  <b
id="key"></b></div><p></p>
    <div id="DetailGrid">
    </div>

```

```

</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Print large number of columns

By default, the browser uses A4 as page size option to print pages and to adapt the size of the page the browser print preview will auto-hide the overflowed contents. Hence grid with large number of columns will cut off to adapt the print page.

To show large number of columns when printing, adjust the scale option from print option panel based on your content size.



Show or Hide columns while Printing

You can show a hidden column or hide a visible column while printing the grid using [toolbarClick](#) and [printComplete](#) events.

In the `toolbarClick` event, based on `args.item.id` as `grid_print`. We can show or hide columns by setting `column.visible` property to `true` or `false` respectively.

In the `printComplete` event, We have reversed the state back to the previous state.

In the below example, we have `CustomerID` as a hidden column in the grid. While printing, we have changed `CustomerID` to visible column and `ShipCity` as hidden column.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.DetailRow, ej.grids.Toolbar, ej.grids.Page);
var grid= new ej.grids.Grid({

```

```

        dataSource: data,
        allowPaging: true,
        toolbar: ['Print'],
        columns: [
            { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
            { field: 'CustomerID', headerText: 'Customer ID', visible:
false, width: 150 },
            { field: 'ShipCity', headerText: 'Ship City', width: 150 },
            { field: 'ShipName', headerText: 'Ship Name', width: 150 }
        ],
        pageSettings: { pageSizes: true, pageSize: 6 },
        toolbarClick : function() {
            for (var i = 0; i < this.columns.length; i++) {
                if (this.columns[i].field == "CustomerID") {
                    this.columns[i].visible = true;
                }
                else if (this.columns[i].field == "ShipCity") {
                    this.columns[i].visible = false;
                }
            }
        },
        printComplete : function() {
            for (var i = 0; i < this.columns.length; i++) {
                if (this.columns[i].field == "CustomerID") {
                    this.columns[i].visible = false;
                }
                else if (this.columns[i].field == "ShipCity") {
                    this.columns[i].visible = true;
                }
            }
        }
    });
    grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Limitations of Printing Large Data

When grid contains large number of data, printing all the data at once is not a best option for the browser performance. Because to render all the DOM elements in one page will produce performance issues in the browser. It leads to browser slow down or browser hang. Grid have option to handle large number of data by Virtualization. However while printing, it is not possible to use virtualization for rows and columns.

If printing of all the data is still needed, we suggest to Export the grid to **Excel** or **CSV** or **Pdf** and then print it from another non-web based application.

See Also

- [How to Print the expanded state grid from all pages](#)
- [How to print only selected records in grid](#)

Adaptive in ##Platform_Name## Grid control

The Grid user interface (UI) was redesigned to provide an optimal viewing experience and improve usability on small screens. This interface will render the filter, sort, column chooser, column

menu(supports only when the `rowRenderingMode` as Horizontal) and edit dialogs adaptively and have an option to render the grid row elements in the vertical direction.

Render adaptive dialogs

When we enable the `enableAdaptiveUI` property, the grid will render the filter, sort, and edit dialogs in full screen for a better user experience. This behavior is demonstrated in the below demo.

INDEX.JS

```
ej.grids.Grid.Inject(ej.grids.Page, ej.grids.Filter, ej.grids.Sort,
ej.grids.Edit, ej.grids.Aggregate, ej.grids.Toolbar);
var grid = new ej.grids.Grid({
  dataSource: data,
  enableAdaptiveUI: true,
  allowPaging: true,
  allowSorting: true,
  allowFiltering: true,
  filterSettings: { type: 'Excel' },
  toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel', 'Search'],
  editSettings: { allowAdding: true, allowEditing: true, allowDeleting:
true, mode: 'Dialog' },
  height: '100%',
  load: function() {
    grid.adaptiveDlgTarget = document.getElementsByClassName('e-mobile-
content')[0];
  },
  columns: [
    { field: 'SNO', headerText: 'S NO', isPrimaryKey: true, width: 150,
validationRules: { required: true, number: true } },
    { field: 'Model', headerText: 'Model Name', width: 200, editType:
"dropdownedit", validationRules: { required: true } },
    { field: 'Developer', headerText: 'Developer', filter: { type :
'Menu' }, width: 200, validationRules: { required: true } },
    { field: 'ReleaseDate', headerText: 'Released Date', type: 'date',
editType: "datepickeredit", format: 'yMMM', width: 200 },
    { field: 'AndroidVersion', headerText: 'Android Version', filter: {
type : 'CheckBox' }, width: 200, validationRules: { required: true } }
  ]
});
grid.appendTo('#adaptivebrowser');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" class="e-adaptive-demo e-bigger">
        <div class="e-mobile-layout">
            <div class="e-mobile-content">
                <div id="adaptivebrowser"></div>
            </div>
        </div>
        <br>
        <div class="datalink">Source:
            <a
href="https://en.wikipedia.org/wiki/List_of_Android_smartphones"
target="_blank">Wikipedia: List of Android smartphones</a>
            </div>
        </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Vertical row rendering

The grid will render the row elements in vertical order while setting the [rowRenderingMode](#) property value as **Vertical**.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Page, ej.grids.Filter, ej.grids.Sort,
ej.grids.Edit, ej.grids.Aggregate, ej.grids.Toolbar);
var grid = new ej.grids.Grid({

```

```

dataSource: data,
enableAdaptiveUI: true,
rowRenderingMode: 'Vertical',
allowPaging: true,
allowSorting: true,
allowFiltering: true,
filterSettings: { type: 'Excel' },
toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel', 'Search'],
editSettings: { allowAdding: true, allowEditing: true, allowDeleting:
true, mode: 'Dialog' },
height: '100%',
load: function() {
    grid.adaptiveDlgTarget = document.getElementsByClassName('e-mobile-
content')[0];
},
columns: [
    { field: 'SNO', headerText: 'S NO', isPrimaryKey: true, width: 150,
validationRules: { required: true, number: true } },
    { field: 'Model', headerText: 'Model Name', width: 200, editType:
"dropdownedit", validationRules: { required: true } },
    { field: 'Developer', headerText: 'Developer', filter: { type :
'Menu' }, width: 200, validationRules: { required: true } },
    { field: 'ReleaseDate', headerText: 'Released Date', type: 'date',
editType: "datepickeredit", format: 'yMMM', width: 200 },
    { field: 'AndroidVersion', headerText: 'Android Version', filter: {
type : 'CheckBox' }, width: 200, validationRules: { required: true } }
],
aggregates: [{
    columns: [{
        type: 'Count',
        field: 'Model',
        footerTemplate: 'Total Models: ${Count}'
    }]
}]
});
grid.appendTo('#verticalrender');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" class="e-adaptive-demo e-bigger">
        <div class="e-mobile-layout">
            <div class="e-mobile-content">
                <div id="verticalrender"></div>
            </div>
        </div>
        <br>
        <div class="datalink">Source:
            <a
href="https://en.wikipedia.org/wiki/List_of_Android_smartphones"
target="_blank">Wikipedia: List of Android smartphones</a>
            </div>
        </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

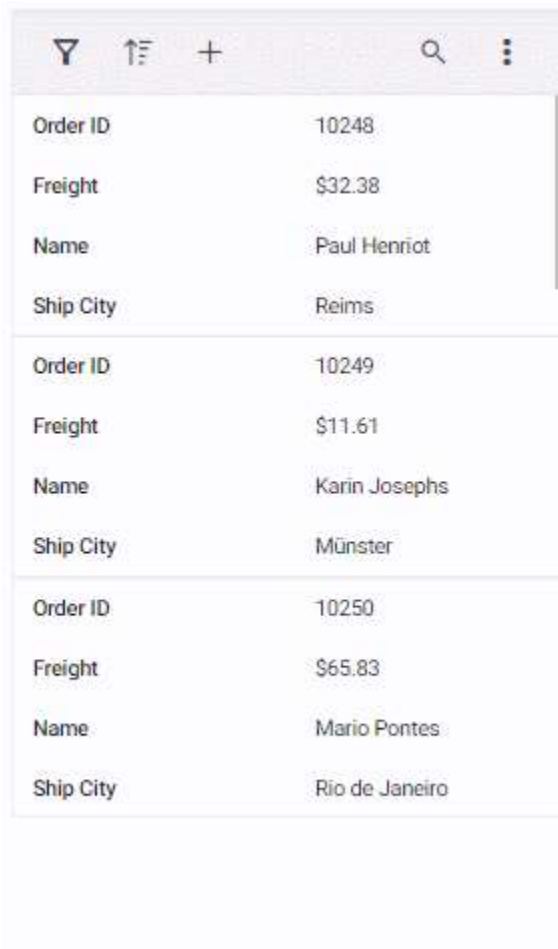
* [enableAdaptiveUI](#) property must be enabled for vertical row rendering.

Supported features by vertical row rendering

The following features are only supported in vertical row rendering:

- Paging, including Page size dropdown
- Sorting
- Filtering
- Selection
- Dialog Editing
- Aggregate

- Infinite scroll
- Toolbar - Options like **Add**, **Filter**, **Sort**, **Edit**, **Delete**, **Search**, and **Toolbar template** are available when their respective features are enabled. The toolbar dynamically includes a three-dotted icon, containing additional features like **ColumnChooser**, **Print**, **PdfExport**, **ExcelExport**, or **CsvExport**, once these features are enabled. Please refer to the following snapshot.



Order ID	10248
Freight	\$32.38
Name	Paul Henriot
Ship City	Reims
Order ID	10249
Freight	\$11.61
Name	Karin Josephs
Ship City	Münster
Order ID	10250
Freight	\$65.83
Name	Mario Pontes
Ship City	Rio de Janeiro

A snapshot of the adaptive grid displaying enabled paging along with a pager dropdown.

<div> <div></div> <div></div> <div></div> </div>	
Order ID	10248
Freight	\$32.38
Name	Vins et alcools Chevalier
Ship City	Reims
Order ID	10249
Freight	\$11.61
Name	Toms Spezialitäten
Ship City	Münster
Order ID	10250
Freight	\$65.83
<div> <div> <div></div> <div></div> <div></div> </div> <div> <div>1 / 6</div> <div></div> <div></div> </div> <div> <div>Items per page</div> <div>5</div> </div> </div>	

The Column Menu feature, which includes grouping, sorting, autofit, filter, and column chooser, is exclusively supported for the Grid in **Horizontal** [rowRenderingMode](#).

Hierarchy grid in ##Platform_Name## Grid control

The Grid allows display of table data in a hierarchical structure to visualize relations between parent and child records. This feature is enabled by defining the [childGrid](#) and [childGrid.queryString](#). The [childGrid](#) describes the options of grid and the [childGrid.queryString](#) describes the relation between parent and child grids.

To use hierarchical binding, inject the [DetailRow](#) module in the grid.

INDEX.JS

```
ej.grids.Grid.Inject(ej.grids.DetailRow);
var grid= new ej.grids.Grid({
  dataSource: employeeData,
  columns: [
    { field: 'EmployeeID', headerText: 'Employee ID', textAlign:
'Right', width: 120 },
    { field: 'FirstName', headerText: 'First Name', width: 150 },
    { field: 'City', headerText: 'City', width: 150 },
    { field: 'Country', headerText: 'Country', width: 150 }
  ],
  childGrid: {
```

```

        dataSource: data,
        queryString: 'EmployeeID',
        columns: [
            { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
            { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
            { field: 'ShipCity', headerText: 'Ship City', width: 150 },
            { field: 'ShipName', headerText: 'Ship Name', width: 150 }
        ],
    },
    height: 315
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
      background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
      background-color: #eee;
    }
  </style>

```

```

        .e-expand::before {
            content: '\e5b8';
        }
        .empImage {
            margin: 6px 16px;
            float: left;
            width: 50px;
            height: 50px;
        }
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

* Grid supports n level of child grids.

* Hierarchical binding is not supported when [DetailTemplate](#) is enabled.

ExpandAll by external button

By default, grid renders in collapsed state. You can expand all child grid rows by invoking the [expandAll](#) method, and collapse all grid rows by invoking the [collapseAll](#) through an external button.

INDEX.JS

```

var expandBtn = new ej.buttons.Button();
expandBtn.appendTo('#expandall');
var collapseBtn = new ej.buttons.Button();
collapseBtn.appendTo('#collapseall');
ej.grids.Grid.Inject(ej.grids.DetailRow);
var grid = new ej.grids.Grid({
    dataSource: employeeData,
    columns: [
        { field: 'EmployeeID', headerText: 'Employee ID', textAlign:
'Right', width: 120 },
        { field: 'FirstName', headerText: 'First Name', width: 150 },
        { field: 'City', headerText: 'City', width: 150 },
        { field: 'Country', headerText: 'Country', width: 150 }
    ],
    childGrid: {
        dataSource: data,

```

```

        queryString: 'EmployeeID',
        columns: [
            { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
            { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
            { field: 'ShipCity', headerText: 'Ship City', width: 150 },
            { field: 'ShipName', headerText: 'Ship Name', width: 150 }
        ],
    },
    height: 265
});
grid.appendTo('#Grid');
document.getElementById('expandall').addEventListener('click', function() {
    grid.detailRowModule.expandAll();
});
document.getElementById('collapseall').addEventListener('click', function()
{
    grid.detailRowModule.collapseAll();
});

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

```



```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="expandall">Expand All</button>
        <button id="collapseall">Collapse All</button>
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Expand child grid initially

You can expand a particular child grid at initial rendering by invoking the [expand](#) method in the [dataBound](#) event.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.DetailRow);
var grid = new ej.grids.Grid({
    dataSource: employeeData,
    columns: [
        { field: 'EmployeeID', headerText: 'Employee ID', textAlign:
'Right', width: 120 },
        { field: 'FirstName', headerText: 'First Name', width: 150 },
        { field: 'City', headerText: 'City', width: 150 },
        { field: 'Country', headerText: 'Country', width: 150 }
    ],
    childGrid: {
        dataSource: data,
        queryString: 'EmployeeID',
        columns: [
            { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
            { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
            { field: 'ShipCity', headerText: 'Ship City', width: 150 },
            { field: 'ShipName', headerText: 'Ship Name', width: 150 }
        ],
    },
    height: 315,
    dataBound: onDataBound
});
grid.appendTo('#Grid');
function onDataBound(){
    this.detailRowModule.expand(1); // initial expand 1 chid Grid.
}

```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
      background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
      background-color: #eee;
    }
    .e-expand::before {
      content: '\e5b8';
    }
    .empImage {
      margin: 6px 16px;
      float: left;
      width: 50px;
      height: 50px;
    }
  </style>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Dynamically load child grid data

You can dynamically load child grid dataSource by using the [load](#) event. This [load](#) event triggers when the child grid is expanded for the first time.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.DetailRow);
var grid = new ej.grids.Grid({
    dataSource: employeeData,
    columns: [
        { field: 'EmployeeID', headerText: 'Employee ID', textAlign:
'Right', width: 120 },
        { field: 'FirstName', headerText: 'First Name', width: 150 },
        { field: 'City', headerText: 'City', width: 150 },
        { field: 'Country', headerText: 'Country', width: 150 }
    ],
    childGrid: {
        queryString: 'EmployeeID',
        columns: [
            { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
            { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
            { field: 'ShipCity', headerText: 'Ship City', width: 150 },
            { field: 'ShipName', headerText: 'Ship Name', width: 150 }
        ]
    },
    detailDataBound: onExpand,
    height: 315
});
grid.appendTo('#Grid');
function onExpand(args) {
    this.childGrid.dataSource = data; // assign data source for child grid.
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>

```

```

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

<div id="container">

```

```

        <div id="Grid"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Bind hierarchy grid with different field

By default, Parent and child grid relation will be maintained by `queryString` property. We should use the same field name to map both parent and child grid. To achieve parent and child relation with different fields, we need to change the mapping value in the child grid `load` event.

In the below sample, we have bound the child and parent grid with different fields. Parent grid field name as `EmployeeID` and the child grid field name as `ID`. We need to define the mapping value of `parentKeyFieldValue` from the parent row data in the child grid `load` event.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.DetailRow);
var grid = new ej.grids.Grid({
    dataSource: employeeData,
    columns: [
        { field: 'EmployeeID', headerText: 'Employee ID', textAlign:
'Right', width: 120 },
        { field: 'FirstName', headerText: 'First Name', width: 150 },
        { field: 'City', headerText: 'City', width: 150 },
        { field: 'Country', headerText: 'Country', width: 150 }
    ],
    childGrid: {
        dataSource: childdata,
        queryString: 'ID',
        columns: [
            { field: 'ID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
            { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
            { field: 'ShipCity', headerText: 'Ship City', width: 150 },
            { field: 'ShipName', headerText: 'Ship Name', width: 150 }
        ],
        load: load,
    },
    height: 315
});
grid.appendTo('#Grid');
function load(args) {
    this.parentDetails.parentKeyFieldValue =
this.parentDetails.parentRowData['EmployeeID'];
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>

```

```

<title>EJ2 Grid</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```

```

<div id="container">
  <div id="Grid"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Adding Record in ChildGrid

Parent and child grid are related by [queryString](#) field value. To maintain this relation in newly added record, You need to set value for [queryString](#) field in the added data by the [actionBegin](#) event.

In the below demo, **EmployeeID** field relates the parent and child grids. To add a new record in child grid, We have to set the **EmployeeID** field with parent record's [queryString](#) field value in the [actionBegin](#) event.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.DetailRow,ej.grids.Edit,ej.grids.Toolbar);
var grid = new ej.grids.Grid({
  dataSource: employeeData,
  columns: [
    { field: 'EmployeeID', headerText: 'Employee ID', textAlign:
'Right', width: 120 },
    { field: 'FirstName', headerText: 'First Name', width: 150 },
    { field: 'City', headerText: 'City', width: 150 },
    { field: 'Country', headerText: 'Country', width: 150 }
  ],
  childGrid: {
    dataSource: data,
    queryString: 'EmployeeID',
    toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
    editSettings: { allowEditing: true, allowAdding: true,
allowDeleting: true },
    columns: [
      { field: 'OrderID', headerText: 'Order ID', isPrimaryKey:true,
textAlign: 'Right', width: 120 },
      { field: 'EmployeeID', headerText: 'Employee ID', textAlign:
'Right', allowEditing:false, width: 120 },
      { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
      { field: 'ShipName', headerText: 'Ship Name', width: 150 }
    ],
    actionBegin: actionBegin
  }
});
grid.appendTo('#Grid');
function actionBegin(args) {
  if (args.requestType === "add") {
    // `parentKeyFieldValue` refers to the queryString field value of
the parent record.
    args.data.EmployeeID = this.parentDetails.parentKeyFieldValue;
  }
}

```

```
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
      background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
      background-color: #eee;
    }
    .e-expand::before {
      content: '\e5b8';
    }
    .empImage {
      margin: 6px 16px;
      float: left;
      width: 50px;
      height: 50px;
    }
  </style>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
```



```

<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Dynamically bind data to child grid based on parent row Data

By default, the [childGrid.queryString](#) describes the relationship between parent and child grids and visualizes the data in a hierarchical structure. Instead of the `queryString` property, we can dynamically bind the datasource to the `childGrid` based on the parent row data using the [detailDataBound](#) event of the grid.

While expanding the child Grid, the `detailDataBound` event will be triggered. In this event, based on the EmployeeID column value of parent row data, filter the equally matched data from the `childGrid` datasource using the `DataManager` plugin and bind the filtered data as a datasource to the `childGrid`. This can be demonstrated by the following sample.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.DetailRow);
var orderData = data;
var grid = new ej.grids.Grid({
    dataSource: employeeData,
    columns: [
        { field: 'EmployeeID', headerText: 'Employee ID', textAlign: 'Right', width: 120 },
        { field: 'FirstName', headerText: 'First Name', width: 150 },
        { field: 'City', headerText: 'City', width: 150 },
        { field: 'Country', headerText: 'Country', width: 150 }
    ],
    childGrid: {
        columns: [
            { field: 'OrderID', headerText: 'Order ID', isPrimaryKey: true, textAlign: 'Right', width: 120 },
            { field: 'EmployeeID', headerText: 'Employee ID', textAlign: 'Right', width: 120 },
            { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
            { field: 'ShipName', headerText: 'Ship Name', width: 150 }
        ],
    },
    detailDataBound: function(args){
        var empIdValue =
args.childGrid.parentDetails.parentRowData.EmployeeID;

```

```

var matchedData = new ej.data.DataManager(orderData).executeLocal(
    new ej.data.Query().where('EmployeeID', 'equal', empIdValue,
true)
);
args.childGrid.query = new ej.data.Query();
args.childGrid.dataSource = matchedData;
}
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
      background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
      background-color: #eee;
    }
    .e-expand::before {
      content: '\e5b8';
    }
    .empImage {

```

```

        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

State Persistence

State persistence in ##Platform_Name## Grid control

State persistence refers to the Grid's state maintained in the browser's [localStorage](#) even if the browser is refreshed or if you move to the next page within the browser. State persistence stores grid's model object in the local storage when the [enablePersistence](#) is defined as true.

Restore initial Grid state

When the [enablePersistence](#) property is set to **true**, the Grid will keep its state even if the page is reloaded. In some cases, you may be required to retain the Grid in its initial state. The Grid will not retain its initial state now since the [enablePersistence](#) property has been enabled.

You can achieve this by destroying the grid after disabling the `enablePersistence` property and clearing the local storage data, as shown in the sample below.

INDEX.JS

```

var grid = new ej.grids.Grid({
    dataSource: data,
    enablePersistence: true,
    allowFiltering: true,
    allowPaging: true,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'ShipCity', headerText: 'Ship City', width: 150 },
        { field: 'ShipName', headerText: 'Ship Name', width: 150 }
    ],
    height: 230

```

```
});
grid.appendTo('#Grid');
document.getElementById('restore').onclick = function () {
    grid.enablePersistence = false;
    window.localStorage.setItem("gridGrid", "");
    grid.destroy();
    //reloads the page
    location.reload();
};
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="restore">Restore to initial state</button>
```

```

        <div id="Grid"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Maintaining custom query in a persistent state

The grid does not maintain the query params after page load event when the [enablePersistence](#) is set to true. This is because the grid refreshes its query params for every page load. You can maintain the custom query params by resetting the [addParams](#) method in the [actionBegin](#) event.

INDEX.JS

```

var grid = new ej.grids.Grid({
    dataSource: data,
    allowFiltering: true,
    allowPaging: true,
    enablePersistence: true,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'ShipCity', headerText: 'Ship City', width: 150 },
        { field: 'ShipName', headerText: 'Ship Name', width: 150 }
    ],
    actionBegin: actionHandler,
    height: 270
});
grid.appendTo('#Grid');
function actionHandler(args) {
    this.query.addParams('$filter', 'EmployeeID eq 1');
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Add a new column in persisted columns list

The Grid columns can be persisted when the [enablePersistence](#) property is set to true. If you want to add the new columns with the existing persist state, you can use the Grid inbuilt method such as `column.push` and call the [refreshColumns\(\)](#) method for UI changes. Please refer to the following sample for more information.

INDEX.JS

```

var grid = new ej.grids.Grid({
    dataSource: data,
    enablePersistence: true,
    allowPaging: true,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'ShipCity', headerText: 'Ship City', width: 150 },

```

```

        { field: 'ShipName', headerText: 'Ship Name', width: 150 }
    ],
    height: 230
});
grid.appendTo('#Grid');
document.getElementById('add').onclick = function () {
    var obj = { field: 'Freight', headerText: 'Freight', width: 120 };
    grid.columns.push(obj);
    grid.refreshColumns();
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">

```

```

        <button id="add">Add columns</button>
        <div id="Grid"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Get or set local storage value in ##Platform_Name## Grid control

If the [enablePersistence](#) property is set to true, the grid property value is saved in the `window.localStorage` for reference. You can get/set the `localStorage` value by using the `getItem/setItem` method in the `window.localStorage`.

```
`ts
```

```
//get the Grid model.
```

```
let value: string = window.localStorage.getItem('gridGrid'); //"gridGrid" is component name +
component id.
```

```
let model: Object = JSON.parse(model);
```

```
,
```

```
`ts
```

```
//set the Grid model.
```

```
window.localStorage.setItem('gridGrid', JSON.stringify(model)); //"gridGrid" is component name +
component id.
```

```
,
```

Prevent to persist in ##Platform_Name## Grid control

Prevent columns from persisting

When the [enablePersistence](#) property is set to true, the Grid properties such as [Grouping](#), [Paging](#), [Filtering](#), [Sorting](#), and [Columns](#) will persist. You can use the `addOnPersist` method to prevent these Grid properties from persisting.

The following example demonstrates how to prevent Grid columns from persisting. In the [dataBound](#) event of the Grid, you can override the `addOnPersist` method and remove the columns from the key list given for persistence.

Note: When the `enablePersistence` property is set to true, the Grid features such as column template, column formatter, header text, and value accessor will not persist.

INDEX.JS

```

var grid = new ej.grids.Grid({
    dataSource: data,
    enablePersistence: true,
    allowPaging: true,
    columns: [

```



```

        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'ShipCity', headerText: 'Ship City', width: 150 },
        { field: 'ShipName', headerText: 'Ship Name', width: 150 }
    ],
    height: 230,
    dataBound: dataBound
});
grid.appendTo('#Grid');
function dataBound(args) {
    var cloned = this.addOnPersist;
    this.addOnPersist = function (key) {
        key = key.filter(function (item) { return item !== "columns"; });
        return cloned.call(this, key);
    };
}
document.getElementById('add').onclick = function () {
    var obj = { field: "Freight", headerText: 'Freight', width: 120 };
    grid.columns.push(obj);
    grid.refreshColumns();
};
document.getElementById('remove').onclick = function () {
    grid.columns.pop();
    grid.refreshColumns();
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="add">Add columns</button>
        <button id="remove">Remove columns</button>
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Add to persist in ##Platform_Name## Grid control

Persist the column template, header template and header Text

By default, the Grid properties such as column template, header text, header template, column formatter, and value accessor will not persist when [enablePersistence](#) is set to true. Because the column template and header text can be customized at the application level.

If you wish to restore all these column properties, then you can achieve it by cloning the grid's columns property using JavaScript Object's assign method and storing this along with the persist data manually. While restoring the settings, this column object must be assigned to the grid's columns property to restore the column settings as demonstrated in the following sample.

INDEX.JS

```

var grid = new ej.grids.Grid({
    dataSource: data,
    enablePersistence: true,
    allowFiltering: true,
    allowPaging: true,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150,
headerTemplate: '#customertemplate' },
        { field: 'ShipCity', headerText: 'Ship City', width: 150 },
        { field: 'ShipName', headerText: 'Ship Name', width: 150, template:
'#template' },

```

```

    ],
    height: 230
});
grid.appendTo('#Grid');
document.getElementById('restore').onclick = function () {
    savedProperties = JSON.parse(grid.getPersistData());
    var gridColumnsState = Object.assign([], grid.getColumns());
    savedProperties.columns.forEach(function (col) {
        var headerText = gridColumnsState.find(function (colColumnsState) {
            return colColumnsState.field === col.field; })['headerText'];
        var colTemplate = gridColumnsState.find(function (colColumnsState) {
            return colColumnsState.field === col.field; })['template'];
        var headerTemplate = gridColumnsState.find(function
        (colColumnsState) { return colColumnsState.field === col.field;
        })['headerTemplate'];
        col.headerText = 'Text Changed';
        col.template = colTemplate;
        col.headerTemplate = headerTemplate;
    });
    console.log(savedProperties);
    grid.setProperties(savedProperties);
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

```

```

<style>
    .e-column:before {
        content: '\e815';
    }
    .e-header:before {
        content: '\ea9a';
    }
</style><script
src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <script id="template" type="text/x-template">
        <a rel='nofollow'
href=https://en.wikipedia.org/wiki/${ShipName}><span class="e-icons e-
column"></span></a>
    </script>

    <script id="customertemplate" type="text/x-template">
        <span class="e-icons e-header" ></span>
        Customer ID
    </script>

    <div id="container">
        <button id="restore">Restore</button>
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Tool Bar

Tool bar in ##Platform_Name## Grid control

The Grid provides ToolBar support to handle grid actions. The [toolbar](#) property accepts either the collection of built-in toolbar items and [ItemModel](#) objects for custom toolbar items or HTML element ID for toolbar template.

To use ToolBar, inject [ToolBar](#) module in the grid.

Enable or disable toolbar items

You can enable/disable toolbar items by using the **enableItems** method.

INDEX.JS

```
ej.grids.Grid.Inject(ej.grids.Toolbar, ej.grids.Group, ej.grids.Sort);
```

```

var grid = new ej.grids.Grid({
  dataSource: data,
  toolbar: ['Expand', 'Collapse'],
  allowGrouping: true,
  groupSettings: { columns: ['CustomerID'] },
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
    { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
    { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C' },
    { field: 'OrderDate', headerText: 'Order Date', textAlign: 'Right',
width: 140, format: 'yMd' }
  ],
  height: 220,
  toolbarClick: clickHandler
});
grid.appendTo('#Grid');
var enable = new ej.buttons.Button({}, '#enable');
var disable = new ej.buttons.Button({}, '#disable');
enable.element.onclick = function() {
  grid.toolbarModule.enableItems(['Grid_Collapse', 'Grid_Expand'],
true); // enable toolbar items.
};
disable.element.onclick = function() {
  grid.toolbarModule.enableItems(['Grid_Collapse', 'Grid_Expand'],
false); // disable toolbar items.
};
function clickHandler(args) {
  if (args.item.id === 'Grid_Collapse') { // grid_Collapse is component id
+ '_' + toolbar item name.
    grid.groupModule.collapseAll();
  }
  if (args.item.id === 'Grid_Expand') {
    grid.groupModule.expandAll();
  }
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="enable" class="e-flat">Enable</button>
        <button id="disable" class="e-flat">Disable</button>
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Add toolbar at the bottom of Grid

You can add the Grid toolbar component at the bottom of Grid using the ['created'](#) event.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Toolbar);
var grid = new ej.grids.Grid({
    dataSource: data.slice(0,8),
    toolbar: ['Print', 'Search'],
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
        { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },

```

```

        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C' },
        { field: 'OrderDate', headerText: 'Order Date', textAlign: 'Right',
width: 140, format: 'yMd' }
    ],
    height: 200,
    created: () => {
        let toolbar = grid.element.querySelector('.e-toolbar');
        grid.element.appendChild(toolbar);
    }
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```

```

    <div id="container">
      <div id="Grid"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Toolbar Component](#)
- [How to set overflow mode in Grid toolbar](#)

Tool bar items in ##Platform_Name## Grid control

Built-in toolbar items

Built-in toolbar items execute standard actions of the grid, and it can be added by defining the [toolbar](#) as a collection of built-in items. It renders the button with icon and text.

The following table shows built-in toolbar items and its actions.

Built-in Toolbar Items	Actions
----- -----	
Add	Adds a new record.
Edit	Edits the selected record.
Update	Updates the edited record.
Delete	Deletes the selected record.
Cancel	Cancels the edit state.
Search	Searches the records by the given key.
Print	Prints the grid.
ExcelExport	Exports the grid to Excel.
PdfExport	Exports the grid to PDF.
WordExport	Exports the grid to Word.
ColumnChooser	Choose the column's visibility.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Toolbar);
var grid = new ej.grids.Grid({
  dataSource: data,
  toolbar: ['Print', 'Search'],
  columns: [

```



```

        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
        { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C' },
        { field: 'OrderDate', headerText: 'Order Date', textAlign: 'Right',
width: 140, format: 'yMd' }
    ],
    height: 270
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
      background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
      background-color: #eee;
    }
    .e-expand::before {

```

```

        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

* The [toolbar](#) has options to define both built-in and custom toolbar items.

Show only icons in built-in toolbar items

By default, the built-in toolbar items render as buttons with an icon and text. It is possible to hide the text and show only the icon using the following CSS style.

```

.e-toolbar .e-tbar-btn-text, .e-toolbar .e-toolbar-items .e-toolbar-item .e-tbar-btn-text {
display: none;
}

```

This is demonstrated in the following sample:

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Toolbar);
var grid = new ej.grids.Grid({
    dataSource: data,
    editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true },
    toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number', isPrimaryKey: true },

```

```

        { field: 'CustomerID', headerText: 'Customer ID', width: 140, type:
'string' },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C' },
        { field: 'OrderDate', headerText: 'Order Date', textAlign: 'Right',
width: 140, format: 'yMd' }
    ],
    height: 270
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <style>
    .e-toolbar .e-tbar-btn-text,
    .e-toolbar .e-toolbar-items .e-toolbar-item .e-tbar-btn-text {
      display: none;
    }
  </style>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>

```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Custom toolbar items

Custom toolbar items can be added by defining the [toolbar](#) as a collection of [ItemModels](#). Actions for this customized toolbar items are defined in the [toolbarClick](#) event.

By default, Custom toolbar items are in position **Left**. You can change the position by using the [align](#) property. In the below sample, we have applied position **Right** for the **Collapse All** toolbar item.

INDEX.JS

```

var clickHandler = function(args){
    if (args.item.id === 'expandall') {
        grid.groupModule.expandAll();
    }
    if (args.item.id === "collapseall") {
        grid.groupModule.collapseAll();
    }
};
ej.grids.Grid.Inject(ej.grids.Toolbar, ej.grids.Group);
var grid = new ej.grids.Grid({
    dataSource: data,
    allowGrouping: true,
    groupSettings: { columns: ['CustomerID'] },
    toolbar: [{ text: 'Expand All', tooltipText: 'Expand All', prefixIcon:
'e-expand', id: 'expandall', align:'left' }, { text: 'Collapse All',
tooltipText: 'collection All', prefixIcon: 'e-collapse', id: 'collapseall',
align:'Right' }],
    toolbarClick: clickHandler,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
        { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C' },
        { field: 'OrderDate', headerText: 'Order Date', textAlign: 'Right',
width: 140, format: 'yMd' }
    ],
    height: 200,
});

```

```
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

  <style>
    .e-icons.e-collapse::before {
      content: "\e834";
    }
    .e-icons.e-expand::before {
      content: "\e83d";
    }
  </style>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Grid"></div>
```

```

    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

* The [toolbar](#) has options to define both built-in and custom toolbar items.

* If a toolbar item does not match the built-in items, it will be treated as a custom toolbar item.

Custom toolbar items are elements that are added to the toolbar, such as input, button, span, etc. In order to focus these items using the TAB or Shift + Tab keys, the `e-toolbar-item-focus` class should be used in the elements. If the `e-toolbar-item-focus` class is not used in the custom toolbar item elements, the focus of the item will be skipped when using the TAB or Shift + Tab keys.

Both built-in and custom items in toolbar

Grid have an option to use both built-in and custom toolbar items at same time.

In the below example, `Add`, `Edit`, `Delete`, `Update`, `Cancel` are built-in toolbar items and `Click` is custom toolbar item.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Edit, ej.grids.Toolbar);
var grid = new ej.grids.Grid({
    dataSource: data,
    toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel', { text: 'Click',
    tooltipText: 'Click', prefixIcon: 'e-expand', id: 'Click' }],
    toolbarClick: clickHandler,
    editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
    true },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
        width: 100, isPrimaryKey: true },
        { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
        width: 120, format: 'C2' },
        { field: 'ShipCountry', headerText: 'Ship Country', width: 150 }
    ],
    height: 273
});
grid.appendTo('#Grid');
function clickHandler(args) {
    if (args.item.id === 'Click') {
        alert("Cutom toolbar click...");
    }
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>

```

```

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

<div id="container">

```

```

        <div id="Grid"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Custom toolbar component in a specific position

By default, Custom toolbar items are in the Left position. You can change the position by using the [align](#) property. In the following sample, we have applied the Right position for the Collapse All toolbar item, Left for the Expand All toolbar item, and Center for the Search toolbar item.

INDEX.JS

```

function toolbarClick(args){
    if (args.item.id === 'expandall') {
        grid.groupModule.expandAll();
    }
    if (args.item.id === "collapseall") {
        grid.groupModule.collapseAll();
    }
};
ej.grids.Grid.Inject(ej.grids.Toolbar, ej.grids.Group);
var grid = new ej.grids.Grid({
    dataSource: employeeData,
    allowGrouping: true,
    groupSettings: { columns: ['FirstName'] },
    toolbar: [{ text: 'Expand All', tooltipText: 'Expand All', prefixIcon:
'e-expand', id: 'expandall', align: 'Left'}, { text: 'Collapse All',
tooltipText: 'collection All', prefixIcon: 'e-collapse', id: 'collapseall',
align: 'Right' }, { text: 'Search', align: 'Center'}],
    toolbarClick: toolbarClick,
    columns: [
        { field: 'EmployeeID', headerText: 'Employee ID', textAlign:
'Right', width: 120, type: 'number' },
        { field: 'FirstName', width: 140, headerText: 'First Name', type:
'string' },
        { field: 'Country', headerText: 'Country', textAlign: 'Right',
width: 120, country: 'string' },
        { field: 'PostalCode', headerText: 'Postal Code', textAlign:
'Right', width: 140, type: 'string' }
    ],
    height: 200,
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">

```



```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

<style>
    .e-icons.e-collapse::before {
        content: "\e834";
    }
    .e-icons.e-expand::before {
        content: "\e83d";
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>

```

```
</body></html>
```

Custom tool bar in ##Platform_Name## Grid control

Custom toolbar is used to customize the whole toolbar. It can be added by defining **toolbarTemplate** as an HTML element ID.

Actions for this toolbar template items are defined in the [toolbarClick](#) event.

INDEX.JS

```
var clickHandler = function(args){
    var target = (args.originalEvent.target).closest('.e-toolbar-item');
    if (target.id === 'collapse') {
        grid.groupModule.collapseAll();
    }
};
ej.grids.Grid.Inject(ej.grids.Toolbar, ej.grids.Group, ej.grids.Sort);
var grid = new ej.grids.Grid({
    dataSource: data,
    toolbarTemplate: '#toolbar-template',
    toolbarClick: clickHandler,
    allowGrouping: true,
    groupSettings: { columns: ['CustomerID'] },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
        { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C2' },
        { field: 'OrderDate', headerText: 'Order Date', textAlign: 'Right',
width: 140, format: 'yMd' }
    ],
    height: 200
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
        <div id="toolbar-template">
            <div>
                <div id="collapse" title="Collapse">
                    <button class="e-btn e-flat">
                        <span class="e-btn-icon e-icons e-collapse"></span>
                    </button>
                </div>
            </div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Render drop-down list in custom toolbar

You can create your own ToolBar items in the Grid. It can be added by defining the [toolbar](#) as HTML element ID. Actions for this ToolBar template items are defined in the [toolbarClick](#).

To include components in the ToolBar, please ensure the following steps:

Step 1:

Initialize the template for your custom component. Using the following code add the DropDownList component to the ToolBar.

```

<div id='toolbar-template'>
<div id='dropdown' style="margin-top:5px">
<input type="text" tabindex="1" id='ddlelement' />
</div>
</div>

```

Step 2:

To render the DropDownList component, use the [dataBound](#) event of the Grid.

- You can select the grid row index based on the selected data in the DropDownList. The output will appear as follows.

INDEX.JS

```

var rowIndex = [0, 1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14];
ej.grids.Grid.Inject(ej.grids.Toolbar);
var grid = new ej.grids.Grid({
  dataSource: data,
  toolbarTemplate: '#toolbar-template',
  dataBound: dataBound,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
    { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
    { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C2' },
    { field: 'OrderDate', headerText: 'Order Date', textAlign: 'Right',
width: 140, format: 'yMd' }
  ],
  height: 200
});
grid.appendTo('#Grid');
function dataBound() {
  var dropDownListObject = new ej.dropdowns.DropDownList({
    // set the data to dataSource property
    dataSource: rowIndex,
    change: change,
    popupHeight :200
  });
  dropDownListObject.appendTo('#ddlelement');
}
function change(args) {
  grid.selectRow(args.itemData);
}

```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
```

```

<title>EJ2 Grid</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="toolbar-template">
            <div id="dropdown" style="margin-top:5px">
                <input type="text" tabindex="1" id="ddlelement">
            </div>
        </div>
        <div id="Grid"></div>
    </div>
<style>
    .orientationcss .e-headercelldiv {
        transform: rotate(90deg);
    }
</style>
<script>
var ele = document.getElementById('container');
if(ele) {

```

```

    ele.style.visibility = "visible";
  }
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Pdf Export

Pdf export in ##Platform_Name## Grid control

PDF export allows exporting Grid data to PDF document. You need to use the [pdfExport](#) method for exporting. To enable PDF export in the grid, set the [allowPdfExport](#) as true.

To use PDF export, inject the PdfExport module in grid.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Toolbar, ej.grids.PdfExport, ej.grids.Page);
var grid = new ej.grids.Grid({
  dataSource: data,
  allowPaging: true,
  pageSettings: { pageSize: 6 },
  allowPdfExport: true,
  toolbar: ['PdfExport'],
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
    { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
    { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C' },
    { field: 'OrderDate', headerText: 'Order Date', textAlign: 'Right',
width: 140, format: 'yMd' }
  ],
  height: 260
});
grid.appendTo('#Grid');
grid.toolbarClick = function(args) {
  if (args['item'].id === 'Grid_pdfexport') {
    let exportProperties = {
      fileName: "new.pdf"
    };
    grid.pdfExport(exportProperties);
  }
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}

```

```
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Show spinner while exporting

You can show/ hide spinner component while exporting the grid using `showSpinner/ hideSpinner` methods. You can use `toolbarClick` event to show spinner before exporting and hide a spinner in the `pdfExportComplete` or `excelExportComplete` event after the exporting.

In the `toolbarClick` event, based on the parameter `args.item.id` as `Gridpdfexport` or `Gridexcelexport` we can call the `showSpinner` method from grid instance.

In the `pdfExportComplete` or `excelExportComplete` event, We can call the `hideSpinner` method.

In the below demo, we have rendered the default spinner component when exporting the grid.

INDEX.JS

```
ej.grids.Grid.Inject(ej.grids.Page, ej.grids.PdfExport,
ej.grids.ExcelExport, ej.grids.Toolbar);
var grid = new ej.grids.Grid({
  dataSource: data,
  allowPaging: true,
  allowPdfExport: true,
  allowExcelExport: true,
  toolbar: ['PdfExport', 'ExcelExport'],
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
    { field: 'CustomerID', width: 140, headerText: 'Customer ID',
visible: false },
    { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120 },
    { field: 'ShipCity', headerText: 'ShipCity', textAlign: 'Right',
width: 140 }
  ],
  height: 260
});
grid.appendTo('#Grid');
grid.toolbarClick = function(args){
  if (args['item'].id === 'Grid_pdfexport') {
    grid.showSpinner();
    grid.pdfExport();
  }
  if (args['item'].id === 'Grid_exceleexport') {
    grid.showSpinner();
    grid.excelExport();
  }
}
grid.pdfExportComplete = () => {
  grid.hideSpinner();
}
grid.excelExportComplete = () => {
  grid.hideSpinner();
}
```


INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
      background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
      background-color: #eee;
    }
    .e-expand::before {
      content: '\e5b8';
    }
    .empImage {
      margin: 6px 16px;
      float: left;
      width: 50px;
      height: 50px;
    }
  </style>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>

```

```

</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Custom data source

PDF export provides an option to define datasource dynamically before exporting. To export data dynamically, define the `dataSource` in `exportProperties`

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Page, ej.grids.PdfExport, ej.grids.Toolbar);
var grid = new ej.grids.Grid({
    dataSource: data,
    allowPaging: true,
    allowPdfExport: true,
    toolbar: ['PdfExport'],
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
        { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C' },
        { field: 'OrderDate', headerText: 'Order Date', textAlign: 'Right',
width: 140, format: 'yMd' }
    ],
    height: 260
});
grid.appendTo('#Grid');
grid.toolbarClick = function(args) {
    if (args['item'].id === 'Grid_pdfexport') {
        let exportProperties = {
            dataSource: data,
        };
        grid.pdfExport(exportProperties);
    }
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

```

```

<meta name="description" content="Typescript Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>

```

```
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Passing additional parameters to the server when exporting

You can pass the additional parameter in the `query` property by invoking `addParams` method. In the `toolbarClick` event, you can define params as key and value pair so it will receive at the server side when exporting.

In the below example, we have passed `recordcount` as `12` using `addParams` method.

INDEX.JS

```
ej.grids.Grid.Inject(ej.grids.Page, ej.grids.PdfExport,
ej.grids.ExcelExport, ej.grids.Toolbar);
var queryClone;
var grid = new ej.grids.Grid({
    dataSource: data,
    allowPaging: true,
    allowPdfExport: true,
    allowExcelExport: true,
    toolbar: ['PdfExport', 'ExcelExport'],
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
        { field: 'CustomerID', width: 140, headerText: 'Customer ID',
visible: false },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120 },
        { field: 'ShipCity', headerText: 'ShipCity', textAlign: 'Right',
width: 140 }
    ],
    height: 260
});
grid.appendTo('#Grid');
grid.toolbarClick = function(args){
    if (args['item'].id === 'Grid_pdfexport') {
        queryClone = grid.query;
        grid.query = new ej.data.Query().addParams("recordcount", "12");
        grid.pdfExport();
    }
    if (args['item'].id === 'Grid_excelexport') {
        queryClone = grid.query;
        grid.query = new ej.data.Query().addParams("recordcount", "12");
        grid.excelExport();
    }
}
grid.pdfExportComplete = () => {
    grid.query = queryClone;
}
grid.excelExportComplete = () => {
```

```
        grid.query = queryClone;
    }
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
      background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
      background-color: #eee;
    }
    .e-expand::before {
      content: '\e5b8';
    }
    .empImage {
      margin: 6px 16px;
      float: left;
      width: 50px;
      height: 50px;
    }
  </style>
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Passing the selected records to the server using ajax request via custom toolbar button click

You can pass the selected records to the server with the help of an ajax request. In the `toolbarClick` event, you can get the selected records using the [getSelectedRecords](#) method and pass the selected records to the server using the **data** property of the ajax.

`ts

```

import { Grid, Page, Toolbar } from '@syncfusion/ej2-grids';
import { Ajax } from '@syncfusion/ej2-base';

Grid.Inject(Page, Toolbar);

let grid: Grid = new Grid({
    dataSource: data,
    allowSelection: true,
    toolbar: ['Selected'],
    columns: [
        { type: 'checkbox', width: 50, },
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right', width: 120},
        { field: 'CustomerID', width: 140, headerText: 'CustomerID'},
        { field: 'ShipCity', headerText: 'Ship Country', width: 140 }
    ],
    toolbarClick: function (args) {
        if (args.item.text === 'Selected') {
            var selectedRecord = this.getSelectedRecords();
            var ajax = new Ajax({

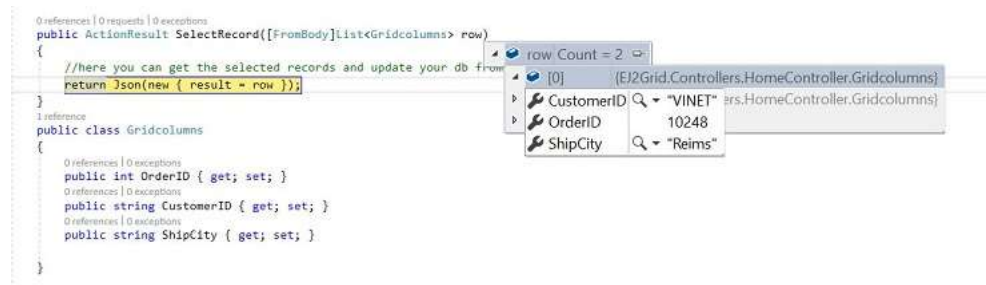
```

```

url: "/Home/SelectRecord",
type: "POST",
contentType: "application/json",
data: JSON.stringify(selectedRecord),
successHandler: function (response) {
    console.log(JSON.parse(response));
},
failure: function (response) {
    alert(response);
}
});
ajax.send();
},
});
grid.appendTo('#Grid');
`

```

The selected record details are bound to the `row` parameter. Please refer to the following screenshot.



[See Also](#)

- [How to Exporting Grid in Cordova application](#)

Export multiple grids in `##Platform_Name##` Grid control

The PDF export provides an option to export multiple grids to the same or different pages of a PDF file. Each grid is identified by its unique ID. You can specify which grid to export by listing their **IDs** in the `exportGrids` property.

[Same page](#)

PDF exporting provides support for exporting multiple grids on the same page. To export the grids on the same page, define `multipleExport.type` as **AppendToPage** in `exportProperties`. It also has an

option to provide blank space between the grids. This blank space can be defined by using `multipleExport.blankSpace` property.

INDEX.JS

```
ej.grids.Grid.Inject(ej.grids.PdfExport, ej.grids.Page, ej.grids.Toolbar);
var firstGrid = new ej.grids.Grid({
  dataSource: data.slice(0, 5),
  allowPaging: true,
  allowPdfExport: true,
  exportGrids: ['FirstGrid', 'SecondGrid'],
  toolbar: ['PdfExport'],
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
    { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
    { field: 'ShipName', headerText: 'Ship Name', width: 150 },
    { field: 'ShipCountry', headerText: 'Ship Country', width: 150 },
  ],
});
firstGrid.appendTo('#FirstGrid');
var secondGrid = new ej.grids.Grid({
  dataSource: employeeData.slice(0, 5),
  allowPaging: true,
  allowPdfExport: true,
  columns: [
    { field: 'EmployeeID', headerText: 'Employee ID', textAlign:
'Right', width: 120, type: 'number' },
    { field: 'FirstName', width: 140, headerText: 'First Name', type:
'string' },
    { field: 'LastName', width: 140, headerText: 'Last Name', type:
'string' },
    { field: 'City', headerText: 'City', textAlign: 'Right', width: 120
},
    { field: 'BirthDate', headerText: 'Birth Date', textAlign: 'Right',
width: 140, format: 'yMd' }
  ],
});
secondGrid.appendTo('#SecondGrid');
firstGrid.toolbarClick = function (args) {
  if (args['item'].id === 'FirstGrid_pdfexport') {
    var appendPdfExportProperties = {
      multipleExport: { type: "AppendToPage", blankSpace: 10 }
    };
    firstGrid.pdfExport(appendPdfExportProperties, true);
  }
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
```



```
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <p><b>First Grid</b></p>
        <div id="FirstGrid"></div>
        <p><b>Second Grid</b></p>
        <div id="SecondGrid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

[New page](#)

PDF export functionality enables the exporting of multiple grids into separate pages (each grid on a new page) within the PDF file. To achieve this, you can specify `multipleExport.type` as **NewPage** in `exportProperties`.

INDEX.JS

```
ej.grids.Grid.Inject(ej.grids.PdfExport, ej.grids.Page, ej.grids.Toolbar);
var firstGrid = new ej.grids.Grid({
  dataSource: data.slice(0, 5),
  allowPaging: true,
  allowPdfExport: true,
  exportGrids: ['FirstGrid', 'SecondGrid'],
  toolbar: ['PdfExport'],
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
    { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
    { field: 'ShipName', headerText: 'Ship Name', width: 150 },
    { field: 'ShipCountry', headerText: 'Ship Country', width: 150 },
  ],
});
firstGrid.appendTo('#FirstGrid');
var secondGrid = new ej.grids.Grid({
  dataSource: employeeData.slice(0, 5),
  allowPaging: true,
  allowPdfExport: true,
  columns: [
    { field: 'EmployeeID', headerText: 'Employee ID', textAlign:
'Right', width: 120, type: 'number' },
    { field: 'FirstName', width: 140, headerText: 'First Name', type:
'string' },
    { field: 'LastName', width: 140, headerText: 'Last Name', type:
'string' },
    { field: 'City', headerText: 'City', width: 120 },
  ],
});
secondGrid.appendTo('#SecondGrid');
firstGrid.toolbarClick = function(args){
  if (args['item'].id === 'FirstGrid_pdfexport') {
    var appendPdfExportProperties = {
      multipleExport: { type: 'NewPage' }
    };
    firstGrid.pdfExport(appendPdfExportProperties, true);
  }
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
```

```

<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <p><b>First Grid</b></p>
        <div id="FirstGrid"></div>
        <p><b>Second Grid</b></p>
        <div id="SecondGrid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Pdf export options in ##Platform_Name## Grid control

[Export current page](#)

PDF export provides an option to export the current page into PDF. To export current page, define the `exportType` to `CurrentPage`.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Page, ej.grids.PdfExport, ej.grids.Toolbar);
var grid = new ej.grids.Grid({
  dataSource: data,
  allowPaging: true,
  allowPdfExport: true,
  toolbar: ['PdfExport'],
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
    { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
    { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C' },
    { field: 'OrderDate', headerText: 'Order Date', textAlign: 'Right',
width: 140, format: 'yMd' }
  ],
  height: 260
});
grid.appendTo('#Grid');
grid.toolbarClick = function(args) {
  if (args['item'].id === 'Grid_pdfexport') {
    var exportProperties = {
      exportType: 'CurrentPage'
    };
    grid.pdfExport(exportProperties);
  }
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Export the selected records only

You can export the selected records data by passing it to `exportProperties.dataSource` Property in the `toolbarClick` event.

In the below exporting demo, We can get the selected records using `getSelectedRecords` method and pass the selected data to `PdfExport` or `excelExport` property.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.PdfExport, ej.grids.ExcelExport,
ej.grids.Page, ej.grids.Toolbar, ej.grids.Filter);
var grid = new ej.grids.Grid({
    dataSource: data,
    allowPdfExport: true,
    allowExcelExport: true,
    allowPaging: true,
    allowFiltering: true,
    selectionSettings: {type: 'Multiple'},
    toolbar: ['PdfExport', 'ExcelExport'],
    pageSettings: { pageCount: 5, pageSize: 5 },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', width: 120,
textAlign: 'Right' },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'OrderDate', headerText: 'Order Date', width: 130,
format: 'yMd', textAlign: 'Right' },
        { field: 'Freight', width: 120, format: 'C2', textAlign: 'Right'
},
        { field: 'ShipCountry', visible: false, headerText: 'Ship
Country', width: 150 }
    ],
});
grid.appendTo('#Grid');
grid.toolbarClick = (args) => {
    if (args.item.id === 'Grid_pdfexport') {
        let pdfdata = grid.getSelectedRecords();
        let exportProperties = {
            dataSource: pdfdata,
        };
        grid.pdfExport(exportProperties);
    }
    else if (args.item.id === 'Grid_excelexport') {
        let exceldata = grid.getSelectedRecords();
        let exportProperties = {
            dataSource: exceldata,
        };
        grid.excelExport(exportProperties);
    }
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>

```

```
</body></html>
```

Export filtered data only

You can export the filtered data by defining the resulted data in `exportProperties.dataSource` before export.

In the below Pdf exporting demo, We have gotten the filtered data by applying filter query to the grid data and then defines the resulted data in `exportProperties.dataSource` and pass it to `pdfExport` method.

INDEX.JS

```
ej.grids.Grid.Inject(ej.grids.PdfExport, ej.grids.Page, ej.grids.Toolbar,
ej.grids.Filter);
var grid = new ej.grids.Grid(
{
    dataSource: data,
    allowPdfExport: true,
    allowPaging: true,
    allowFiltering: true,
    toolbar: ['PdfExport'],
    pageSettings: { pageCount: 5, pageSize: 5 },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', width: 120,
textAlign: 'Right' },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'OrderDate', headerText: 'Order Date', width: 130,
format: 'yMd', textAlign: 'Right' },
        { field: 'Freight', width: 120, format: 'C2', textAlign: 'Right'
},
        { field: 'ShipCountry', visible: false, headerText: 'Ship
Country', width: 150 }
    ],
});
grid.appendTo('#Grid');
grid.toolbarClick = function(args) {
    if (args.item.id === 'Grid_pdfexport') {
        let pdfdata;
        let query = grid.renderModule.data.generateQuery(); // get grid
corresponding query
        for (var i = 0; i < query.queries.length; i++) {
            if (query.queries[i].fn == 'onPage') {
                query.queries.splice(i, 1); // remove page query to get
all records
                break;
            }
        }
        let fdata = new ej.data.DataManager({ json: data
}).executeQuery(query).then((e) => {
            pdfdata = e.result; // get all filtered records
            let exportProperties = {
                dataSource: pdfdata,
            };
            grid.pdfExport(exportProperties);
        }).catch((e) => true);
    }
}
```



```
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Grid"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
<script src="index.js" type="text/javascript"></script>
```

```
</body></html>
```

Export hidden columns

PDF export provides an option to export hidden columns of Grid by defining the `includeHiddenColumn` as `true`.

INDEX.JS

```
ej.grids.Grid.Inject(ej.grids.Page, ej.grids.PdfExport, ej.grids.Toolbar);
var grid = new ej.grids.Grid({
  dataSource: data,
  allowPaging: true,
  allowPdfExport: true,
  toolbar: ['PdfExport'],
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
    { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
    { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C' },
    { field: 'ShipCountry', width: 140, headerText: 'Ship Country',
visible: false },
    { field: 'OrderDate', headerText: 'Order Date', textAlign: 'Right',
width: 140, format: 'yMd' }
  ],
  height: 260
});
grid.appendTo('#Grid');
grid.toolbarClick = function(args) {
  if (args['item'].id === 'Grid_pdfexport') {
    var exportProperties = {
      includeHiddenColumn: true
    };
    grid.pdfExport(exportProperties);
  }
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Show or hide columns

You can show a hidden column or hide a visible column while exporting the grid using [toolbarClick](#) and [pdfExportComplete](#) events.

In the `toolbarClick` event, based on `args.item.id` as `Grid_pdfexport`. We can show or hide columns by setting `column.visible` property to `true` or `false` respectively.

In the `pdfExportComplete` event, We have reversed the state back to the previous state.

In the below example, we have `CustomerID` as a hidden column in the grid. While exporting, we have changed `CustomerID` to visible column and `ShipCity` as hidden column.

INDEX.JS

```
ej.grids.Grid.Inject(ej.grids.Page, ej.grids.PdfExport, ej.grids.Toolbar);
var grid = new ej.grids.Grid({
  dataSource: data,
  allowPaging: true,
  allowPdfExport: true,
  toolbar: ['PdfExport'],
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
    { field: 'CustomerID', width: 140, headerText: 'Customer ID',
visible: false },
    { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120 },
    { field: 'ShipCity', headerText: 'ShipCity', textAlign: 'Right',
width: 140 }
  ],
  height: 260
});
grid.appendTo('#Grid');
grid.toolbarClick = function(args) {
  if (args['item'].id === 'Grid_pdfexport') {
    grid.columns[1].visible = true;
    grid.columns[3].visible = false;
    grid.pdfExport();
  }
}
grid.pdfExportComplete = () => {
  grid.columns[1].visible = false;
  grid.columns[3].visible = true;
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}

```

```

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Change page orientation

Page orientation can be changed Landscape(Default Portrait) for the exported document using the `exportProperties`.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Page, ej.grids.PdfExport, ej.grids.Toolbar);
var grid = new ej.grids.Grid({
  dataSource: data,
  allowPaging: true,
  allowPdfExport: true,
  toolbar: ['PdfExport'],
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
    { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
    { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C' },
    { field: 'OrderDate', headerText: 'Order Date', textAlign: 'Right',
width: 140, format: 'yMd' }
  ],
  height: 260
});
grid.appendTo('#Grid');
grid.toolbarClick = function(args) {
  if (args['item'].id === 'Grid_pdfexport') {
    let exportProperties = {
      pageOrientation: 'Landscape',
    };
    grid.pdfExport(exportProperties);
  }
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Change page size

Page size can be customized for the exported document using the `exportProperties`.

Supported page sizes are:

- Letter
- Note
- Legal
- A0
- A1
- A2
- A3
- A5
- A6
- A7
- A8
- A9
- B0
- B1
- B2
- B3
- B4
- B5
- Archa
- Archb
- Archc
- Archd
- Arche
- Flsa
- HalfLetter
- Letter11x17
- Ledger

INDEX.JS

```
ej.grids.Grid.Inject(ej.grids.Page, ej.grids.PdfExport, ej.grids.Toolbar);
var grid = new ej.grids.Grid({
  dataSource: data,
  allowPaging: true,
  allowPdfExport: true,
  toolbar: ['PdfExport'],
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
    { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
    { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C' },
    { field: 'OrderDate', headerText: 'Order Date', textAlign: 'Right',
width: 140, format: 'yMd' }
  ],
```



```

        height: 260
    });
    grid.appendTo('#Grid');
    grid.toolbarClick = function(args) {
        if (args['item'].id === 'Grid_pdfexport') {
            var exportProperties = {
                pageSize: 'Letter'
            };
            grid.pdfExport(exportProperties);
        }
    }
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <style>
        .e-row[aria-selected="true"] .e-customizedExpandcell {
            background-color: #e0e0e0;
        }
        .e-grid.e-gridhover tr[role='row']:hover {
            background-color: #eee;
        }
        .e-expand::before {
            content: '\e5b8';
        }
    </style>

```

```

    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Define file name

You can assign the file name for the exported document by defining `fileName` property in [PdfExportProperties](#).

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Toolbar, ej.grids.PdfExport, ej.grids.Page);
var grid = new ej.grids.Grid({
    dataSource: data,
    allowPaging: true,
    pageSettings: { pageSize: 6 },
    allowPdfExport: true,
    toolbar: ['PdfExport'],
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
        { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C' },
        { field: 'OrderDate', headerText: 'Order Date', textAlign: 'Right',
width: 140, format: 'yMd' }
    ],
    height: 260
});
grid.appendTo('#Grid');
grid.toolbarClick = function(args){
    if (args['item'].id === 'Grid_pdfexport') {

```

```

        let exportProperties = {
            fileName: "new.pdf"
        };
        grid.pdfExport (exportProperties);
    }
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <style>
        .e-row[aria-selected="true"] .e-customizedExpandcell {
            background-color: #e0e0e0;
        }
        .e-grid.e-gridhover tr[role='row']:hover {
            background-color: #eee;
        }
        .e-expand::before {
            content: '\e5b8';
        }
        .empImage {
            margin: 6px 16px;
            float: left;
            width: 50px;

```

```

        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Font customization

Default fonts

By default, grid uses Helvetica font in the exported document. You can change the default font by using `pdfExportProperties.theme` property. The available default fonts are,

- Helvetica
- TimesRoman
- Courier
- Symbol
- ZapfDingbats

The code example for changing default font,

```
`ts
```

```
import { PdfStandardFont, PdfFontFamily, PdfFontStyle } from '@syncfusion/ej2-pdf-export';
```

```
...
```

```
let pdfExportProperties: PdfExportProperties = {
```

```
  theme: {
```

```
    header: {font: new PdfStandardFont(PdfFontFamily.TimesRoman, 11, PdfFontStyle.Bold),
```

```
    caption: { font: new PdfStandardFont(PdfFontFamily.TimesRoman, 9) },
```

```
    record: { font: new PdfStandardFont(PdfFontFamily.TimesRoman, 10) }
```

```
  }
```

```
};
```

Add custom font

You can change the default font of Grid header, content and caption cells in the exported document by using `pdfExportProperties.theme` property.

In the following example, we have used Advent Pro font to export the grid with Hungarian fonts.

INDEX.JS

```
ej.grids.Grid.Inject(ej.grids.PdfExport, ej.grids.Page, ej.grids.Toolbar,
ej.grids.Group);
var grid = new ej.grids.Grid({
  dataSource: data,
  allowPaging: true,
  allowPdfExport: true,
  allowGrouping: true,
  toolbar: ['PdfExport'],
  columns: [
    { field: 'OrderID', headerText: 'Rendelés azonosító', textAlign:
'Right', width: 120, type: 'number' },
    { field: 'CustomerID', width: 140, headerText: 'Ügyfél-azonosító',
type: 'string' },
    { field: 'Freight', headerText: 'fuvar', textAlign: 'Right', width:
120, format: 'C' },
    { field: 'OrderDate', headerText: 'Rendelés dátuma', textAlign:
'Right', width: 140, format: 'yMd' }
  ],
  height: 260
});
grid.appendTo('#Grid');
grid.toolbarClick= function(args){
  if (args['item'].id === 'Grid_pdfexport') {
    var pdfExportProperties = {
      theme: {
        header: {font: new
ej.pdfexport.PdfTrueTypeFont(window.adventProFont, 12) },
        caption: { font: new
ej.pdfexport.PdfTrueTypeFont(window.adventProFont, 10) },
        record: { font: new
ej.pdfexport.PdfTrueTypeFont(window.adventProFont, 9) }
      }
    };
    grid.pdfExport(pdfExportProperties);
  }
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

PdfTrueTypeFont accepts base 64 format of the Custom Font.

Export grid as blob

The Grid offers an option to export the data as a **Blob** instead of downloading it as a file in the browser. To export the grid as a Blob, set the **isBlob** parameter to **true** in the [pdfExport](#) method. The grid returns the promise of a blob in the [pdfExportComplete](#) event.

The following example demonstrates how to obtain the blob data of the exported grid by executing the promise in the `pdfExportComplete` event.

INDEX.JS

```
ej.grids.Grid.Inject(ej.grids.Toolbar, ej.grids.PdfExport, ej.grids.Page);
var grid = new ej.grids.Grid({
  dataSource: data,
  allowPaging: true,
  allowPdfExport: true,
  toolbar: ['PdfExport'],
  toolbarClick: toolbarClick,
  pdfExportComplete: pdfExportComplete,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
    { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
    { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C' },
    { field: 'OrderDate', headerText: 'Order Date', textAlign: 'Right',
width: 140, format: 'yMd' }
  ],
  height: 230
});
grid.appendTo('#Grid');
function toolbarClick(args) {
  if (args.item.id === 'Grid_pdfexport') {
    // pass fourth parameter as true to get the blob data of exported
grid
    grid.pdfExport(null, null, null, true);
  }
}
function pdfExportComplete(args) {
  // execute the promise to get the blob data
  args.promise.then((e) => {
    console.log(e.blobData);
  });
};
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Pdf cell style customization in ##Platform_Name## Grid control

Conditional cell formatting

Grid cells in the exported PDF can be customized or formatted using [pdfQueryCellInfo](#) event. In this event, we can format the grid cells of exported PDF document based on the column cell value.

In the below sample, we have set the background color for **Freight** column in the exported document by `args.cell` and `backgroundColor` property.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Page, ej.grids.PdfExport, ej.grids.Toolbar);
var grid = new ej.grids.Grid({
    dataSource: data,
    allowPaging: true,
    allowPdfExport: true,
    toolbar: ['PdfExport'],
    columns: [

```



```

        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
        { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120 },
        { field: 'OrderDate', headerText: 'Order Date', textAlign: 'Right',
width: 140, format: 'yMd' }
    ],
    height: 260
});
grid.appendTo('#Grid');
grid.toolbarClick = function(args){
if (args['item'].id === 'Grid_pdfexport') {
    grid.pdfExport();
}
}
grid.pdfQueryCellInfo = (args) => {
    if(args.column.field == 'Freight'){
        if(args.value < 30) {
            args.style = {backgroundColor: '#99ffcc'};
        }
        else if(args.value < 60) {
            args.style = {backgroundColor: '#ffffb3'};
        }
        else {
            args.style = {backgroundColor: '#ff704d'};
        }
    }
}
grid.queryCellInfo = (args) => {
    if(args.column.field == 'Freight'){
        if(args.data['Freight'] < 30) {
            args.cell.bgColor = '#99ffcc';
        }
        else if(args.data['Freight'] < 60) {
            args.cell.bgColor = '#ffffb3';
        }
        else {
            args.cell.bgColor = '#ff704d';
        }
    }
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}

```

```
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Theme

PDF export provides an option to include theme for exported PDF document.

To apply theme in exported PDF, define the **theme** in **exportProperties**.

INDEX.JS

```
ej.grids.Grid.Inject(ej.grids.Page, ej.grids.PdfExport, ej.grids.Toolbar);
var grid = new ej.grids.Grid({
  dataSource: data,
  allowPaging: true,
  allowPdfExport: true,
  toolbar: ['PdfExport'],
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
    { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
    { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C' },
    { field: 'OrderDate', headerText: 'Order Date', textAlign: 'Right',
width: 140, format: 'yMd' }
  ],
  height: 260
});
grid.appendTo('#Grid');
grid.toolbarClick = function(args) {
  if (args['item'].id === 'Grid_pdfexport') {
    var exportProperties = {
      theme: {
        header: {
          fontColor: '#64FA50', fontName: 'Calibri', fontSize: 17,
bold: true, border: { color: '#64FA50', lineStyle: 'Thin' }
        },
        record: {
          fontColor: '#64FA50', fontName: 'Calibri', fontSize: 17,
bold: true
        },
        caption: {
          fontColor: '#64FA50', fontName: 'Calibri', fontSize: 17,
bold: true
        }
      }
    };
    grid.pdfExport(exportProperties);
  }
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
<title>EJ2 Grid</title>
```

```
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

<div id="container">
```

```

        <div id="Grid"></div>
    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

By default, material theme is applied to exported PDF document.

Adding header and footer in `##Platform_Name##` Grid control

You can customize text, page number, line, page size and changing orientation in header and footer.

Write a text in header and footer

You can add text either in Header or Footer of exported PDF document.

```
`ts
```

```

let exportProperties: PdfExportProperties = {
header: {
fromTop: 0,
height: 130,
contents: [
{
type: 'Text',
value: "Northwind Traders",
position: { x: 0, y: 50 },
style: { textBrushColor: '#000000', fontSize: 13 }
},
]
}
,

```

Draw a line in header and footer

you can add line either in Header or Footer of the exported PDF document.

Supported line styles:

- dash
- dot
- dashdot
- dashdotdot
- solid

```
`ts
let exportProperties: PdfExportProperties = {
  header: {
    fromTop: 0,
    height: 130,
    contents: [
      {
        type: 'Line',
        style: { penColor: '#000080', penSize: 2, dashStyle: 'Solid' },
        points: { x1: 0, y1: 4, x2: 685, y2: 4 }
      }
    ]
  }
}
```

Add page number in header and footer

you can add page number either in Header or Footer of exported PDF document.

Supported page number types:

- LowerLatin - a, b, c,
- UpperLatin - A, B, C,
- LowerRoman - i, ii, iii,
- UpperRoman - I, II, III,
- Number - 1,2,3.

```
`ts
let exportProperties: PdfExportProperties = {
  header: {
    fromTop: 0,
    height: 130,
    contents: [
      {
        type: 'PageNumber',
        pageNumberType: 'Arabic',
        format: 'Page { $current } of { $total }', //optional
        position: { x: 0, y: 25 },
```

```
style: { textBrushColor: '#ffff80', fontSize: 15, hAlign: 'Center' }
}
]
}
}
}
```

Insert an image in header and footer

Image (Base64 string) can be added in the exported document in header/footer using the `exportProperties`.

```

`ts
let exportProperties: PdfExportProperties = {
  header: {
    fromTop: 0,
    height: 130,
    contents: [
      {
        type: 'Image',
        src: image,
        position: { x: 40, y: 10 },
        size: { height: 100, width: 250 },
      }
    ]
  }
}
`

```

The below code illustrates the pdf export customization.

INDEX.JS

```
ej.grids.Grid.Inject(ej.grids.PdfExport, ej.grids.Page, ej.grids.Toolbar);
var image =
"/9j/4AAQSkZJRgABAQAAQABAAD/2wBDAAUDBAQEAwUEBAQFBUQUGBwwIBwcHBw8LCwkMEQ8SEhE
PERETFhwXExQaFRERGCEYGh0dHx8fExcjCjIEjBweHx7/2wBDAAQUFBQcGBw4ICA4eFBEUUh4eHh4
eHh4eHh4eHh4eHh4eHh4eHh4eHh4eHh4eHh4eHh4eHh4eHh4eHh4eHh4eHh4eHh4eHh4eHh4eHh7/wAARCA
DAfAOTDASI
AAhEBAxEB/8QAHQABAAIDAQEBAQAAAAAAAAAAcIBQYJBAMBAv/EAE8QAAECBAEECwoKCAyDAAA
AAAAABAgMEBQYHCBFwSxIYITY3QXR1l1LLSFhcXNVFVYZWj0xMUFSIjMlRikBzEzQlNxc5O00UNSGaH
B8GNY4f/EABsBAQAwEBAIAAAAAAAAAAAAAAAAAABgEDBQIh/8QAOREAAQMBawgJAwIHAQAAAAAAAAAE
CAwQFEVESEyExMkFcxkYUFTM0UMGBsAHB0SLwFjVTVLh8SP/2gAMAAEAAHEDEQA/Al1gAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAHgr9WlKLS4s/OPzMZuNanlnu4mp6V/+8R4kkZEExXvW5E1qemMd15G
ts9VPeDS7KvyXrEf4jUmQ50cc5fgdivzIiz9xqKvgdxeni8huhHog6CtizrLr0+OJuggSWlkzcgX
```

KAASyOAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAD8e5rGK97ka1qZ1VVzIiAHynpqXkZOLNzcVs
KBCbsnvd4EQg687imLiqixnbKHKws7ZeCq/VT/Mv3l4/w/fkcRLrdXZz4nJvVKbAd83/zOT9Zfr5
E/wBf3akfMuklu9cflEbF/NNa+Zfw7HXgXywrI6s3PSP+tfon5/5iCRrDv1YWwptejKrPqwpty7
rfIj/AEfe/Hykcg4Vn2jPQS5yFeKbl4nXrKKKsjzcqe+9OBZJqo5qOaqKipnRU4z9IcsW9ZiiKyR
n9nMU5VzJxvgf+v1b938PIsvSczLzkrDmpWMyNAiN2THsXOiofVLKtiC0o8pmhya03p+U9T57aNm
S0L7n6UXUUP8Av0PqADrHOAAABrmJF5UuxLVjXHWIM5G1IMSHDcyVY10T09yNTMjnNTwr5TYyIMs
LgMqPLJTXNN1MxJJWsdqVUPLluaqmJ20uH3mm5uiwffDbS4feabm6LB98U5BY+yafBeZFz7i4221
w+803N0WD74baXD7zTc3RYPvinIHZNPGvMZ9xcbbS4feabm6LB98fSWyoLamJmFLspVyo6LEaxqr
LQc2dVzJn+19JTU9VG8cSPKYXXQwtk092peYSZx0xIcuJKNsi3rkqNBnqbcL5qnzD5eK6DLwlYrm
rmVWqsVFzfvRCYznnjZww3dztH6ynIs2mqHuR+5DfK9WpoLL7aXD7zTc3RYPvhtpcPvNNzdFg++
Kcg7HZNPGvM0Z9xcbbS4feabm6LB98NtLh95pubosh3xTkDsmnwXmM+4uNtpcPvNNzdFg++JDwpX
GoeI9KnaLQpaoS8GTmPi8RJyGxjldsWuzpsXO3MzkOexbbIW3i3FzumohkOus+GGFXs1nuOVznXK
WFABwiSCLsULt+MuiUKmxPoGrsZqKlfrqn6iehOPy+Dy58tibdvYfCfRqbFzTsRv00Rq7sFq8Sfe
VPwTd40InTcTMhROk9u3X0cC/5L9vzyxLdYfKX3VMYf4p9/xzwAAKEXAAAAGes+6J63Zn6JVjSb1
zxZdy7i/eb5Hfnx8WbAg3U9RLTSJLEtzk3muaG0dixyJeilhaHVpGsyDZ2QjJEhruOTwOY7ja5OJ
T3EG4fOrSXJBZRH7GI79Pskzw/g0XdV6eTd3OPou54Scj6xYVqutKnzj23KmhCF4fvQfOrXs9tDN
kNdei6fVOIAB2jlAiDLC4DKjyyU1zSXyIMsLgMqPLJTXNJNF4hnFPk8P2VKRGaURAAAAB6qN44ke
Uwuuh5T1UbxxI8phddDC6jKHTe5542cMN3c7R+sp0MOeeNnDDd3O0frKV6xe8dwJU+pDUAAWIiAA
AAttKlBxbi53TUQypJbbIW3i3FzumohnOtXwy+3ybYdssKACqk0ifEGypmTjR6vTfhZmWiOWJHhq
quiQ1Vc6uz+Fzf909KbqaIWTI7vywmxvhKnQoTWxfrRZVNxH+VWeRfR4F4sy+Gg290YVL6ikt1Vv
3T8csC42PbyLdDURwX8/nniRgD9clzXK1zVa5q51RUzKipxKfhrs2gAAA+8hKTM/OwpOUhLFjxnB
FjE41/wCE41XiQ+LGue9rGNc971RGtamdVVFaiJxqTNh3ajaFJ/G5trXVKO35/H8E3w7BP+V8v7j
r2NZMlpT5CaGprXBPYu7mc607RZQxZS6XLqT97jI2bbsvbtLSXYqRjMjmdMRs313eRPupxJ/yqmb
APrdPBHTxpFglzU1HzaaZ8z1ket6qAAbjWCIMsLgMqPLJTXNJfIgywuAyo8slNc0k0XiGcU+Tw/Z
UpGAC5EAAAAHqo3jiR5TC66H1PVRvHEjymF10MLqModMTnnjZww3dztH6ynQw5542cMN3c7R+spX
rF7x3AlT6kNQABYiIAAAC22QtvFuLndNRDKkltshbeLcXO6aiGc61fDL7fJth2ywoAKqTQAADUL6
suXrbXTsjsJepIm67wNjZuJ3p+9OfiiGclZiTmokrNwXwI8Ncz2PTMqL/wB4+MsaYK77YkbiLub
G+hm4aZoMw1N1voXyt9H4ZipW70aZV3z0+h+9Nzvwrv34ljjsi3XU10U+lmO9P9ftMCCR4Eznvrt
In6LPukqhB+DiJutcm62I3/MleNP+qbbhhafx+M2tVKFnlIa55eG5P0rk/WX7qf7r6E3aHSWbUVV
T1ZrnbN792KqW+proYIM+q3t3Xb+BlsL7S+KsZXKnCzTD0zy0Jyfo2r+uv314vInpXcKEA+uWfQR
UECQxaklriuKnzetrJKyVZP+JgAATSKAAACKMrCQn6ldgvPylMkJufmXTcqrYMrAdFiKiRmqqlq
KuZE3SVwbIZM1I1+C3mFS9LjnB3GXlodcvqiY7A7jLy0OuX1RMdg6Pg6/bTvJ9TR1dMTnB3GXlod
cvqiY7A7jLy0OuX1RMdg6Pgdt08n1HV0xOcHcZeWhly+qJjsHqpFnXi2rStNwfcjWpMw1VVPmwiI
mzTdx5h0VAW2neT6jq6YgoZjHallzWLF1TMratfMIEWqRnw4sGmR3selXLmVrkaqKnpQvmDn0dWt
K5XI195tezLS45wdx15aHXL6omOwO4y8tDrl9UTHYoj4Oh207yfU1dXTE5wdx15aHXL6omOwO4y8
tDrl9UTHYoj4HbTvJ9R1dMTnB3GXlodcvqiY7BabIrpVVPn11+DVqVUKbFiVRHsZOSr4LnN+Bhpn
RHoiqmdFTOnkJ5BhqbTdPGsatuPTiclb7wADmG4AAAAA8NbpFPrMokrUZZseGjtk3OqorV8qKm6
h7IUNkKE2FCY1kNjUalrUzI1E8CIh/QPCRMR6vRevXWu/QelkcrUYq6E3AAHs8AAAAAAijKynp6
nYLT81Tp6bkZhs3KokaWjuhPRFjNRURzVRcyoSURBlhcBlR5ZKa5pJo0vqGcU+Ty/ZUP73X3dpdc
frWP2x3X3dpdcfrWP2zCguGQ3AgXma7r7u0uuP1rH7Y7r7u0uuP1rH7ZhQMhuAvM13X3dpdcfrWP
2z00i7btdVpJrrsuJzVmYaKi1WOqKmzTcX55rh6qN44keUwuuhhWNU1BFOMJQRGS6LolsWbql5a5
67LwIVUjthwoVSjMYxqOXmJWo7MiehC+pzzxs4Ybu52j9ZSv2MiLI6/AlT6kMV3X3dpdcfrWP2x3
X3dpdcfrWP2zCgsOQ3Ai3ma7r7u0uuP1rH7Y7r7u0uuP1rH7ZhQMhuAvM13X3dpdcfrWP2y0+RVV
KpVLKr8Wq1SfqERlVRrHzcy+M5rfgYa5kV6qqJnVVzekp8W2yFt4txc7pqIZz7Ua1KZbkW+TbCv6
yUsX5iYlralny0xGgPWcaihRFYqpsH7mdP3EV/K1W861DpT/AO5J+NG9eV5czVxXCD4N0qle20F
RFXUh9K6OxsdRIqpvU9nytVvOtQ6U/wDuPlaredah0p/9zxgrmfk8y8zu5qPyPyPZ8rVbZrU01P8
A7j5Wq3nWodKf/c8YGfk8y8xmo/KnIzNv1SqPr9NY+pzzmunIKOa6YeqKivTOipnJ6K9W7vipnLY
OsaWFL/0fe50UuUbt+1Cm9KGNbJHcl2hQAC6FWAAAAAABEGWfWGVHlkprmkvkQZYXAZUeWSmaSaL
xDOKfJ4fsUjAbciAAAD1UbxxI8phddDyngq3jiR5TC66GF1GUOmJzzxs4Ybu52j9ZT0Yc88bOG
G7udo/WUR1i947Sp9SGoAAsREAAABbbIW3i3FzumohLSS22QtvFuLndNRD0davl9vk2w7ZJuNG
9eV5czVxCJCW8aN68ry5mriESHwLpZ/MV4IfTujngk4qAAVo7oAAB7rd3xUzlsHWNLClerd3xUz1
sHWNLCn0HoT3UvFCmdKu8j4KAAXCqgAAAAAIGywuAyo8slNc0l8iDLC4DKjyyU1zSTReIZxT5PD
9lSkYALkQAAAAeqjeOJH1MLroeU9VG8cSPKYXXQwuoyh0xOeeNnDDd3O0frKdDDnnjZww3dztH6y
lesXvHcCVPqQ1AAFiIgAAALbZC28W4ud01EMqSW2yFt4txc7pqIZzrV8Mvt8m2HbJNxo3ryvLmau
IRIS3jRvXleXM1cQiQ+BdLP5ivBD6d0c8EnFQACThDAAPdbu+Kmctg6xpYUr1bu+Kmctg6xpYU+
g9Ce6l4oUzpV3kfBQAC7lUAAAAABruItuUq+rWjW7WYk1Dk40SHEc6Wejh52ORyZ1VFTwp5DYjV
cV7117AsuYuaakI09CgRYUNYMJ6NcqvejEXOu5uZ85siR6vTI17jC3XaSONq7h19uuLpcP3Y2ruH


```

X264ulw/dmC22FE0OqnSoY22FE0OqnSoZ1Mi0fXmhpviM7tXcOvt1xdLh+7G1dw6+3XF0uH7swW2
womh1U6VDG2womh1U6VDGRaPrzQXxGd2ruHX264ulw/dn0lsmPD2XmYUdk9cKvhPa9uebh5s6LnT
/AA/Qa9tsKJodV0lQz+5fKsokWPDhJZ9URXvRuf4zD3M65jGRaPrzQXxFiyH7nydrFuG46hXZ6cr
rZqfmHzEZIUzDRiOcudcyLDXMn+pMBBF6ZS1Jtm7apb0alqjMRKdMul3RWTDEa9W8aIu6hCpUnVy
5jWbH5N36j7bV3Dr7dcXS4fuxtXcOvt1xdLh+7MftsKJodV0lQxtsKJodV0lQydkWj680Nd8Rndq
7h19uuLpcP3Y2ruHX264ulw/dmC22FE0OqnSoY22FE0OqnSoYyLR9eaC+Izuldw6+3XF0uH7skHC
zDqhYc0qcp1BjT8WDNzHxiIs3Fa9yO2KN3FRrdzM1CiDthRNDqp0qGSjgtiZJ4mUioVGTpUzTmyU
yku5kaI16uVWI7Ombi3TRUNrEjXO35J6asd/6TabmoUnX5BknOvjNhsipFRYtKrc6IqcaLufOU17
vZ2/+3qP81vZP3GPEGVw3taBXpumx6hDjTjJVIUGIjHirmPdss68XzP8AciXbYUTQ6qdKhnIf0fi
rlzzoUcuJOitOembkMkVEJZ72dv8A7eo/zW9kd7O3/wBvUf5reyRNTsKJodV0lQxtsKJodV0lQzz
/AAjT/wBun0NnblX/AFVJZ72dv/t6j/Nb2R3s7f8A29R/mt7JE22womh1U6VDG2womh1U6VDH8I0
/9un0HblX/VUL6Sw6oUpOQJqHGn1fAitiNR0VqpnaqKmf5voNwK/UDKgolWr9NpLLTqUJ8/OQZVs
R0zDVGLEejEcvoRXZywJsjsplnfpZGjL8DRNWylSossq4AA2GkAAAAAAEQ5YHAXU+VymvaS8Rdl
gcBdT5XKa9pJovEM4p8niTYUpEAC5EAAAAH3p3jCW/jM6yHwPvTvGET/GZ1kMLqModNDnvjpwYXb
zpF/M6EHPfHThku3nSL+ZXrF713D7kqfZNMABYiIAAAC2eQrvLuTnVupYVMLZ5Cu8u5OdW6lhZrV
8Mvt8m2HbMplucEtP57g6mMU4Lj5bnBLT+e4OpjFODfk+H91MzbQAB0jSAAAZzDvhEtfnuS/qGHR
85wYd8Ilr89yX9Qw6PlftraZ7kqn1KAACQkAAAAAAiHLA4C6nyuU17SXiIcsDgLfK5TXtJNF4h
nFPk8SbClIgAXIgAAAA+9O8YS38ZnWQ+B96d4wlv4zOshhdRldpoc98dOGS7edIv5nQg5746cMl2
86RfzK9Yveu4fclT7JpgALERAAAawzyFd5dyc6t1LCphbPIV3l3Jzq3UsOdavhl9vk2w7ZlMtZgl
p/PcHUXinBcfLc4Jafz3B1MypwYsnw/upmbaAAokaQAADOYd8Ilr89yX9Qw6PnODDvhEtfnuS/qG
HR8r9tbTPclU+pQADiEgAAAAAAEQ5YHAXU+VymvaS8RnlOUGsXJhDP0qg0+NUJ6JMyz2QIWBZORs
ZquXdVE3ERVJFIqJOxVxT5PD9lSiAn77zmKOhFU9n2h3nMUdCKp7PtFu6xF505oQsl2BogN77zmK
OhFU9n2h3nMUdCKp7PtDrEXnTmgyXYGiH3p3jCW/jM6yG6d5zFHQiqez7R9ZHB/E9k7Ae+yqojWx
Wqq/R7iIqfeMLURXbac0CNDgX7Oe+OnDJDvOkX8zoQUoxewsxEquKNy1Om2jUZqTmahEiQIzNhsX
tVdxUzuznAsd7WSOVy3aPuSZ0VU0ENg3vvOYo6EVT2faHecxR0Iqns+0WDrEXnTmhGyXYGiA3vvO
Yo6EVT2faHecxR0Iqns+0OsRedOaDjdgaIWzyFd5dyc6t1LCB+85ijoRVPZ9osjkg2pcdqWrXZa5
KPM0uNMVFsSEyNsc72fBNTOmZV40VDn2nNG6nVGURdW/1NsLVR2o+WW5ws0/nuDqYxTgu5lZ23Xr
ow2kqdbtLj1KbZVoUZ0KDm2SMSFFRXbqpuZ3J+JV7vOYo6EVT2faMWXLG2nuc5E0rvEzVV2hDRAB
33nMUdCKp7PtDvOYo6EVT2faOjl1LzpzQ1ZLsDRAB33nMUdCKp7PtDvOYo6EVT2faHWIvOnNBkuw
MBh3wiWvz3Jf1DDo+UWsjCXEqSve352as2pwpeXqspGjRHLdZMY2MxznL87wiIkpek4VsSMe5uSq
KSYEVEW8AA4xvAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAP/9k=";

var grid = new ej.grids.Grid({
    dataSource: data,
    allowPaging: true,
    allowPdfExport: true,
    toolbar: ['PdfExport'],
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
        { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C' },
        { field: 'OrderDate', headerText: 'Order Date', textAlign: 'Right',
width: 140, format: 'yMd' }
    ],
    height: 260
});
grid.appendTo('#Grid');
grid.toolbarClick= function(args){
    if (args['item'].id === 'Grid_pdfexport') {
        var pdfExportProperties = {
            header: {
                fromTop: 0,
                height: 130,
                contents: [
                    {

```

```

        type: 'Image',
        src: image,
        position: { x: 40, y: 10 },
        size: { height: 100, width: 250 },
    }
    ],
},
footer: {
    fromBottom: 160,
    height: 150,
    contents: [
        {
            type: 'PageNumber',
            pageNumberType: 'Arabic',
            format: 'Page {$current} of {$total}',
            position: { x: 0, y: 25 },
            style: { textBrushColor: '#ffff80', fontSize: 15 }
        }
    ]
}
};
grid.pdfExport(pdfExportProperties);
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Repeat column header on every page

By default, column header will be placed on the first page of the pdf document but you can display column header on every page using `repeatHeader` property of `pdfGrid`.

In the below sample, we have enabled `repeatHeader` property in `pdfHeaderQueryCellInfo` event to show the header on every page.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Page, ej.grids.PdfExport, ej.grids.Toolbar);
var grid = new ej.grids.Grid({
    dataSource: purchaseData,
    allowPaging: true,
    allowPdfExport: true,
    toolbar: ['PdfExport'],

```

```

        columns: [
            { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120},
            { field: 'CustomerID', width: 140, headerText: 'Customer ID'},
            { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C' },
            { field: 'ShipName', headerText: 'ShipName', textAlign: 'Right',
width: 140 }
        ],
        height: 260
    });
    grid.appendTo('#Grid');
    grid.toolbarClick = function(args) {
        if (args['item'].id === 'Grid_pdfexport') {
            grid.pdfExport();
        }
    }
    grid.pdfHeaderQueryCellInfo = function(args){
        args.cell.row.pdfGrid.repeatHeader = true;
    }

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

```

```

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Exporting hierarchy grid in ##Platform_Name## Grid control

The grid have an option to export the hierarchy grid to pdf document. By default, grid will exports the master grid with expanded child grids alone. you can change the exporting option by using the PdfExportProperties.hierarchyExportMode property. The available options are,

Mode	Behavior
Expanded	Exports the master grid with expanded child grids.
All	Exports the master grid with all the child grids.
None	Exports the master grid alone.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.DetailRow, ej.grids.Toolbar,
ej.grids.PdfExport);
var grid= new ej.grids.Grid({

```

```

        dataSource: employeeData,
        toolbar: ["PdfExport"],
        allowPdfExport: true,
        columns: [
            { field: 'EmployeeID', headerText: 'Employee ID', textAlign:
'Right', width: 120 },
            { field: 'FirstName', headerText: 'First Name', width: 150 },
            { field: 'City', headerText: 'City', width: 150 },
            { field: 'Country', headerText: 'Country', width: 150 }
        ],
        childGrid: {
            dataSource: data,
            queryString: 'EmployeeID',
            columns: [
                { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
                { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
                { field: 'ShipCity', headerText: 'Ship City', width: 150 },
                { field: 'ShipName', headerText: 'Ship Name', width: 150 }
            ],
        },
        toolbarClick: toolbarClick
    });
    grid.appendTo('#Grid');
    function toolbarClick(args) {
        if (args['item'].id === 'Grid_pdfexport') {
            var exportProperties = {
                hierarchyExportMode: "Expanded"
            };
            grid.pdfExport(exportProperties);
        }
    }
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Exporting grid with templates in ##Platform_Name## Grid control

The grid offers the option to export the column, detail, and caption templates to a PDF document. The template contains images, hyperlinks, and customized text.

Exporting with column template

The PDF export functionality allows you to export Grid columns that include images, hyperlinks, and custom text to an PDF document.

In the following sample, the hyperlinks and images are exported to PDF using [hyperlink](#) and [image](#) properties in the [pdfQueryCellInfo](#) event.

PDF Export supports base64 string to export the images.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Toolbar, ej.grids.PdfExport);
var grid = new ej.grids.Grid({
    dataSource: employeeData,
    allowPdfExport: true,
    toolbar: ['PdfExport'],
    columns: [
        {
            headerText: 'Employee Image', textAlign: 'Center',

```

```

        template: '#imageTemplate', width: 150
    },
    { field: 'EmployeeID', headerText: 'Employee ID', textAlign:
'Right', width: 125 },
    { field: 'FirstName', headerText: 'Name', width: 120 },
    { field: 'EmailID', headerText: 'Email ID', template:
'#mailTemplate', width: 170 }
    ],
    toolbarClick: toolbarClick,
    pdfQueryCellInfo: pdfQueryCellInfo,
    height: 273
});
grid.appendTo('#Grid');
function toolbarClick(args) {
    if (args.item.id === 'Grid_pdfexport') {
        grid.pdfExport();
    }
}

function pdfQueryCellInfo(args) {
    if (args.column.headerText === 'Employee Image') {
        args.image = {
            base64: args.data.EmployeeImage,
            height: 50,
            width: 50,
        };
    }
    if (args.column.headerText === 'Email ID') {
        args.hyperLink = {
            target: 'mailto:' + args.data.EmailID,
            displayText: args.data.EmailID,
        };
    }
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">

```







```
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <script id="imageTemplate" type="text/x-template">
            <div class="image">
                
            </div>
        </script>
        <script id="mailTemplate" type="text/x-template">
            <div class="link">
                <a href="mailto:${EmailID}">${EmailID}</a></div>
            </script>
            <div id="Grid"></div>
        </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

PDF Export		
Employee Image	Name	Email ID
	Nancy	nancy@domain.com
	Andrew	andrew@domain.com
	Janet	janet@domain.com
	Margaret	margaret@domain.com

Exporting with detail template

By default, the grid will export the parent grid with expanded detail rows alone. Change the exporting option by using the `PdfExportProperties.hierarchyExportMode` property. The available options are:

- | Mode | Behavior |
- |-----|-----|
- | Expanded | Exports the parent grid with expanded detail rows. |
- | All | Exports the parent grid with all the detail rows. |
- | None | Exports the parent grid alone. |

The detail rows in the exported PDF can be customized or formatted using the [exportDetailTemplate](#) event. In this event, the detail rows of the PDF document are formatted in accordance with their parent row details.

In the following sample, the detail row content is formatted by specifying the [columnCount](#), [columnHeader](#), and [rows](#) properties using its [parentRow](#) details. This allows for the creation of detail rows in the PDF document. Additionally, custom styles can be applied to specific cells using the [style](#) property.

If `columnCount` is not provided, the columns in the detail row of the PDF grid will be generated based on the count of the `columnHeader/rows` first row's [cells](#).

When using [rowSpan](#), it is essential to provide the cell's [index](#) for proper functionality.

INDEX.JS

```
ej.grids.Grid.Inject(ej.grids.DetailRow, ej.grids.Toolbar,
ej.grids.PdfExport);
var grid= new ej.grids.Grid({
  dataSource: employeeData,
  detailTemplate: '#detailtemplate',
  toolbar: ['PdfExport'],
  allowPdfExport: true,
  toolbarClick: toolbarClick,
  exportDetailTemplate: exportDetailTemplate,
  columns: [
    { field: 'Category', headerText: 'Category', width: 140 },
    { field: 'ProductID', headerText: 'Product ID', width: 120 },
    { field: 'Status', headerText: 'Status', width: 120 }
  ],
  height: 273
});
grid.appendTo('#Grid');
function toolbarClick(args) {
  if (args['item'].id === 'Grid_pdfexport') {
    var exportProperties = {
      hierarchyExportMode: "Expanded"
    };
    grid.pdfExport(exportProperties);
  }
}
function exportDetailTemplate(args) {
  args.value = {
    columnCount: 2,
    columnHeader: [
      {
        cells: [{
          index: 0, colSpan: 2, value: 'Product Details',
          style: { backgroundColor: '#ADD8E6', pdfTextAlignment:
'Center', bold: true }
        }]
      },
    ],
    rows: [
      {
        cells: [
          {
            index: 0, rowspan: 4, image: { base64:
args.parentRow.data['ProductImg'], width: 80 }
          },
          {
            index: 1, value: "Offers: " +
args.parentRow.data['Offers'],
            style: { fontColor: '#0A76FF', fontSize: 15 }
          },
        ],
      },
      {
        cells: [
```

```

        {
            index: 1, value: 'Available: ' +
args.parentRow.data['Available']
        }
    },
    {
        cells: [
            {
                index: 1, value: 'Contact: ',
                hyperlink: {
                    target: 'mailto:' +
args.parentRow.data['Contact'],
                    displayText: args.parentRow.data['Contact']
                }
            }
        ]
    },
    {
        cells: [
            {
                index: 1, value: 'Ratings: ' +
args.parentRow.data['Ratings'],
                style: { fontColor: '#0A76FF', fontSize: 15 }
            }
        ]
    },
    {
        cells: [
            {
                index: 0, value: args.parentRow.data['productDesc'],
                style: { pdfTextAlignment: 'Center' }
            },
            { index: 1, value: args.parentRow.data['ReturnPolicy'] }
        ]
    },
    {
        cells: [
            {
                index: 0, value: args.parentRow.data['Cost'],
                style: { bold: true, pdfTextAlignment: 'Center' }
            },
            { index: 1, value: args.parentRow.data['Cancellation'] }
        ]
    },
    {
        cells: [
            {
                index: 0, value: args.parentRow.data['Status'],
                style: {
                    fontColor: args.parentRow.data['Status'] ===
'Available' ? '#00FF00' : '#FF0000',
                    pdfTextAlignment: 'Center', fontSize: 15
                }
            },
            {
                index: 1, value: args.parentRow.data['Delivery'],
                style: { fontColor: '#0A76FF', fontSize: 15 }
            }
        ]
    }

```

```

    ]
    }
    ],
    };
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <script id="detailtemplate">
    <table class="detailtable" width="100%">
      <colgroup>
        <col width="40%" />
        <col width="60%" />
      </colgroup>
      <thead>

```

```

        <tr>
            <th colspan="2" style="font-weight: 500;text-align:
center;background-color: #ADD8E6;">
                Product Details
            </th>
        </tr>
    </thead>
    <tbody>
        <tr>
            <td rowspan="4" style="text-align: center;">
                
            </td>
            <td>
                <span style="font-weight: 500;color:
#0a76ff;">Offers: {Offers} </span>
            </td>
        </tr>
        <tr>
            <td>
                <span>Available: {Available} </span>
            </td>
        </tr>
        <tr>
            <td>
                <span class="link">
                    Contact: <a
href="mailto:{Contact}">{Contact}</a>
                </span>
            </td>
        </tr>
        <tr>
            <td>
                <span style="font-weight: 500;color: #0a76ff;">
Ratings: {Ratings}</span>
            </td>
        </tr>
        <tr>
            <td style="text-align: center;">
                <span> {productDesc}</span>
            </td>
            <td>
                <span>{ReturnPolicy}</span>
            </td>
        </tr>
        <tr>
            <td style="text-align: center;">
                <span style="font-weight: 500;" > {Cost}</span>
            </td>
            <td>
                <span>{Cancellation}</span>
            </td>
        </tr>
        <tr>
            <td style="text-align: center;">
                <span class="{Status}" style="font-weight: 500;" >
{Status}</span>

```

```

        </td>
        <td>
            <span style="font-weight: 500;color:
#0a76ff;">${Delivery}</span>
        </td>
    </tr>
</tbody>
</table>
</script>
<div id="container">
    <div id="Grid"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

PDF Export			
	Category	Product ID	Status
▶	Suits/Slim	EJ-SU-01	Available
▶	Suits/Classic	EJ-SU-02	Available
▶	Suits/Formal	EJ-SU-03	Available
▶	Phants/Slim	EJ-PH-01	Available
▶	Phants/Classic	EJ-PH-02	Available
▶	Shirts/Slim	EJ-SH-01	Available
▶	Shirts/Formal	EJ-SH-02	Available

Exporting with caption template

The PDF export feature enables exporting of Grid with a caption template to an PDF document.

In the following sample, the customized caption text is exported to PDF using [captionText](#) property in the [exportGroupCaption](#) event.

INDEX.JS

```
ej.grids.Grid.Inject(ej.grids.Group, ej.grids.Toolbar, ej.grids.PdfExport);
var grid = new ej.grids.Grid({
  dataSource: employeeData,
  allowGrouping: true,
  groupSettings: { captionTemplate: '#captiontemplate', columns:
  ['EmployeeID'] },
  allowPdfExport: true,
  toolbar: ['PdfExport'],
  columns: [
    { field: 'EmployeeID', headerText: 'Employee ID', width: 120 },
    { field: 'FirstName', headerText: 'Name', width: 120 },
    { field: 'City', headerText: 'City' },
    { field: 'Title', headerText: 'Title', width: 170 }
  ],
  toolbarClick: toolbarClick,
  exportGroupCaption: exportGroupCaption,
  height: 273
});
grid.appendTo('#Grid');
function toolbarClick(args) {
  if (args.item.id === 'Grid_pdfexport') {
    grid.pdfExport();
  }
}

function exportGroupCaption(args) {
  args.captionText = args.data.field + ' - ' + args.data.key;
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
```



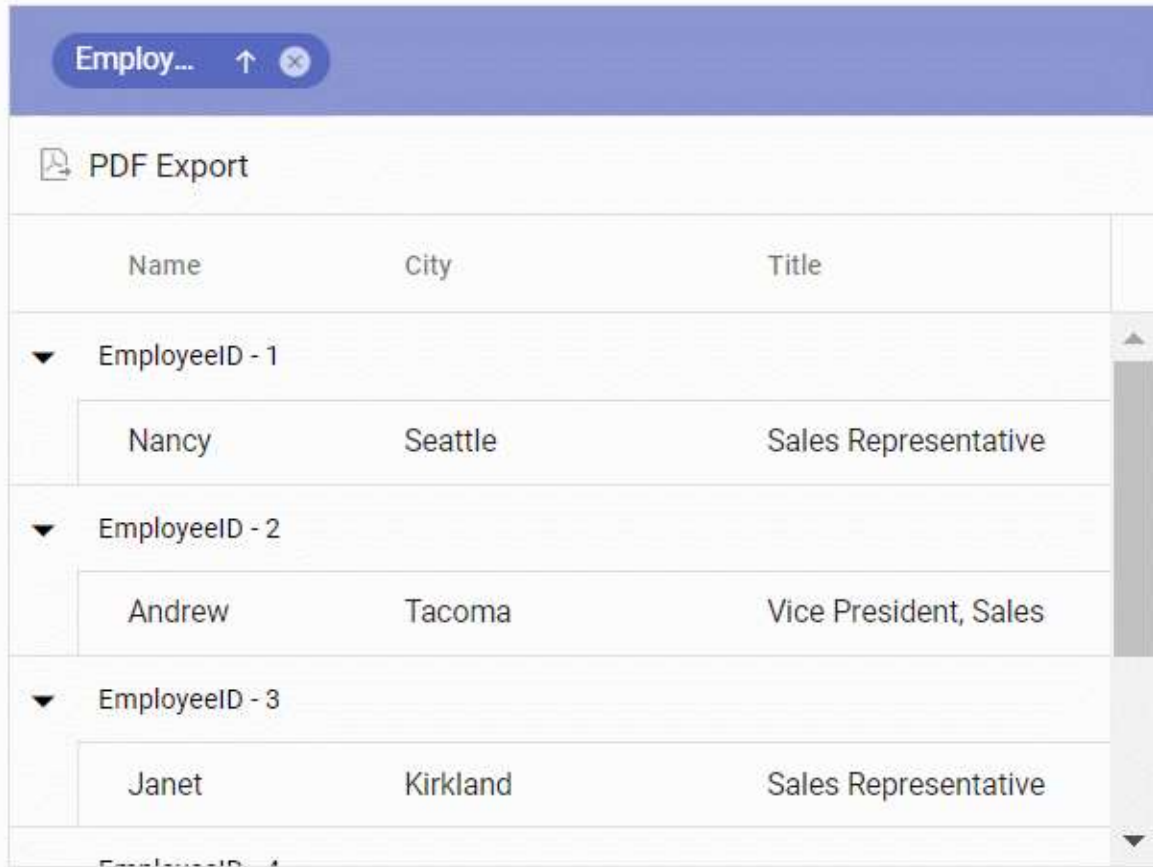
```
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <script id="captiontemplate" type="text/x-template">
            ${field} - ${key}
        </script>
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```



Name	City	Title
EmployeeID - 1		
Nancy	Seattle	Sales Representative
EmployeeID - 2		
Andrew	Tacoma	Vice President, Sales
EmployeeID - 3		
Janet	Kirkland	Sales Representative

Exporting grid in server in ##Platform_Name## Grid control

The Grid have an option to export the data to PDF in server side using Grid server export library.

Server dependencies

The Server side export functionality is shipped in the Syncfusion.EJ2.GridExport package, which is available in Essential Studio and [nuget.org](https://www.nuget.org). The following list of dependencies is required for Grid server side PDF exporting action.

- Syncfusion.EJ2
- Syncfusion.EJ2.GridExport
- Syncfusion.Compression.Base
- Syncfusion.Pdf.Base

Server configuration

The following code snippets shows server configuration using ASP.NET MVC Controller Action.

To Export the Grid in server side, You need to call the

[serverPdfExport](#) method for passing the Grid properties to server exporting action.

```
`ts
public ActionResult PdfExport(string gridModel)
{
```

```

GridPdfExport exp = new GridPdfExport();
Grid gridProperty = ConvertGridObject(gridModel);
return exp.PdfExport<OrdersDetails>(gridProperty, OrdersDetails.GetAllRecords());
}

private Grid ConvertGridObject(string gridProperty)
{
    Grid GridModel = (Grid)Newtonsoft.Json.JsonConvert.DeserializeObject(gridProperty, typeof(Grid));
    GridColumnModel cols =
    (GridColumnModel)Newtonsoft.Json.JsonConvert.DeserializeObject(gridProperty,
    typeof(GridColumnModel));
    GridModel.Columns = cols.columns;
    return GridModel;
}

public class GridColumnModel
{
    public List<GridColumn> columns { get; set; }
}

public ActionResult DataSource(DataManager dm)
{
    var DataSource = OrderRepository.GetAllRecords();
    DataResult result = new DataResult();
    result.result = DataSource.Skip(dm.Skip).Take(dm.Take).ToList();
    result.count = result.result.Count;
    return Json(result, JsonRequestBehavior.AllowGet);
}
`ts

import { Grid, Toolbar } from '@syncfusion/ej2-grids';
import { DataManager, UrlAdaptor } from '@syncfusion/ej2-data';
Grid.Inject(Toolbar);
let data: DataManager = new DataManager({
    url: "Home/DataSource",
    adaptor: new UrlAdaptor
});

```

```

let grid: Grid = new Grid({
  dataSource: data,
  toolbar: ['PdfExport'],
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right', width: 100 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
    { field: 'Freight', headerText: 'Freight', textAlign: 'Right', width: 120, format: 'C2' },
    { field: 'ShipCountry', headerText: 'Ship Country', width: 150 }
  ],
  height: 265
});
grid.appendTo('#Grid');
grid.toolbarClick = (args: Object) => {
  if (args['item'].id === 'Grid_pdfexport') {
    grid.serverPdfExport("Home/PdfExport");
  }
}

```

Note: Refer to the GitHub sample for quick implementation and testing from [here](#).

Export grid as memory stream

The Grid offers an option to export the data as a memory stream instead of downloading it as a file in the browser. To obtain the memory stream of the exported grid, set the **AsMemoryStream** parameter to **true** in the [PdfExport](#) method.

The following code demonstrates how to get the memory stream of exported grid.

```

`ts
public object PdfExport(string gridModel)
{
  GridPdfExport exp = new GridPdfExport();
  Grid gridProperty = ConvertGridObject(gridModel);
  // pass third parameter as true to get the Memory Stream of exported grid data
  return (MemoryStream)exp.PdfExport<OrdersDetails>(gridProperty, OrdersDetails.GetAllRecords(),
  true);
}

```

Merge grid's memory stream

The [Essential PDF](#) library is used to merge multiple memory streams into a single stream. To learn more about the merge option, please refer to this [documentation](#).

You can merge a memory stream, a file stream, and a local file with the Grid's memory stream in the following ways:

Merging with an existing memory stream

If you already have a memory stream, you can directly use it to merge with the Grid's memory stream.

In the following code, the [Merge](#) method of the [PdfDocumentBase](#) class is used to merge the grid's memory stream with an existing memory stream.

```
`ts
using Syncfusion.Pdf;

public MemoryStream ms1; // defines existing memory stream
public object PdfExport(string gridModel)
{
    GridPdfExport exp = new GridPdfExport();
    Grid gridProperty = ConvertGridObject(gridModel);
    // get the memory stream of exported grid data
    MemoryStream ms2 = (MemoryStream)exp.PdfExport<OrdersDetails>(gridProperty,
    OrdersDetails.GetAllRecords(), true);
    //Creates a PDF document.
    PdfDocument finalDoc = new PdfDocument();
    //Creates a PDF stream for merging. ms1 and ms2 represents the existing stream and grid's stream.
    Stream[] streams = { ms1, ms2 };
    //Merges PDFDocument.
    PdfDocumentBase.Merge(finalDoc, streams);
    //Save the document into stream.
    MemoryStream ms3 = new MemoryStream();
    finalDoc.Save(ms3);
    ms3.Position = 0;
    //Close the document.
    finalDoc.Close(true);
    //Dispose the streams.
    ms1.Dispose();
    ms2.Dispose();
    return ms3;
```

```
}
`ts
```

Merging with an existing file stream

If you already have a file stream, you can directly use it to merge with the Grid's memory stream. In the following code, the existing file stream is merged with the Grid's memory stream.

```
`ts
using Syncfusion.Pdf;

public FileStream fs1; // defines existing file stream

public ActionResult PdfExport(string gridModel)
{
    GridPdfExport exp = new GridPdfExport();
    Grid gridProperty = ConvertGridObject(gridModel);
    MemoryStream ms1 = (MemoryStream)exp.PdfExport<OrdersDetails>(gridProperty,
    OrdersDetails.GetAllRecords(), true);
    PdfDocument finalDoc = new PdfDocument();
    //fs1 and ms1 represents the existing stream and grid's stream.
    Stream[] streams = { fs1, ms1 };
    PdfDocumentBase.Merge(finalDoc, streams);
    MemoryStream ms3 = new MemoryStream();
    finalDoc.Save(ms3);
    ms3.Position = 0;
    return ms3;
}
`ts
```

Merging with a local file

To merge a local file with the Grid's memory stream, you need to convert it into a file stream before merging. In the following code, the existing local file is merged with the Grid's memory stream.

```
`ts
using Syncfusion.Pdf;

// get the file stream of local file

public FileStream fs1 = new FileStream("D:/PdfDoc.pdf", FileMode.Open, FileAccess.Read); //
PdfDoc.pdf is a local file which is located in local disk D.

public ActionResult PdfExport(string gridModel)
{
    GridPdfExport exp = new GridPdfExport();
```

```

Grid gridProperty = ConvertGridObject(gridModel);
MemoryStream ms1 = (MemoryStream)exp.PdfExport<OrdersDetails>(gridProperty,
OrdersDetails.GetAllRecords(), true);
PdfDocument finalDoc = new PdfDocument();
//fs1 and ms1 represents the local file's stream and grid's stream.
Stream[] streams = { fs1, ms1 };
PdfDocumentBase.Merge(finalDoc, streams);
MemoryStream ms3 = new MemoryStream();
finalDoc.Save(ms3);
ms3.Position = 0;
return ms3;
}
`

```

Downloading the merged memory stream

You can download the merged memory stream by converting it into a `FileStreamResult`. In the following code, the merged memory stream is downloaded to the browser.

```

`ts
using Syncfusion.Pdf;
public ActionResult PdfExport(string gridModel)
{
    PdfDocument finalDoc = new PdfDocument();
    //ms1 and ms2 represents the streams needs to merge.
    Stream[] streams = { ms1, ms2 };
    PdfDocumentBase.Merge(finalDoc, streams);
    MemoryStream ms3 = new MemoryStream();
    finalDoc.Save(ms3);
    ms3.Position = 0;
    // Save the MemoryStream into FileStreamResult
    FileStreamResult fileStreamResult = new FileStreamResult(ms3, "application/pdf");
    fileStreamResult.FileName = "Export.pdf";
    //Close the document.
    finalDoc.Close(true);
    //Dispose the streams.
    ms1.Dispose();
}
`

```

```
ms2.Dispose();
// return the file
return fileStreamResult;
}
`
```

Rotate a header text to a certain degree in the exported grid on the server side

The Grid has support to customize the column header styles such as changing text orientation, the font color, and so on in the exported PDF file. To achieve this requirement, define the `BeginCellLayout` event of the `PdfExportProperties` with an event handler to perform the required action.

The `PdfHeaderQueryCellInfoEvent` will be triggered when creating a column header for the pdf document to be exported. Collect the column header details in this event and handle the custom in the `BeginCellLayout` event handler.

In the following demo, the `DrawString` method from the `Graphics` is used to rotate the header text of the column header inside the `BeginCellLayout` event handler.

A PDF exporting is not supported to rotate the column header on the client side.

```
`ts
public ActionResult PdfExport(string gridModel)
{
    GridPdfExport exp = new GridPdfExport();
    Grid gridProperty = ConvertGridObject(gridModel);
    gridProperty.ServerPdfHeaderQueryCellInfo = PdfHeaderQueryCellInfo;
    PdfGrid grid = new PdfGrid();
    PdfExportProperties pdfExportProperties = new PdfExportProperties();
    pdfExportProperties.IsRepeatHeader = true;
    pdfExportProperties.BeginCellLayout = new PdfGridBeginCellLayoutEventHandler(BeginCellEvent);
    gridProperty.ServerPdfQueryCellInfo = PdfQueryCellInfo;
    IEnumerable data = Utils.DataTableToJson(dt);
    var result = exp.PdfExport<dynamic>(gridProperty, data, pdfExportProperties);
    return result;
}

public void BeginCellEvent(object sender, PdfGridBeginCellLayoutEventArgs args)
{
    PdfGrid grid = (PdfGrid)sender;
    var brush = new PdfSolidBrush(new PdfColor(Color.DimGray));
    args.Graphics.Save();
}
```



```

args.Graphics.TranslateTransform(args.Bounds.X + 50, args.Bounds.Height + 40); // give the value for
bounds x and Y by the user

args.Graphics.RotateTransform(-60); // give the rotate degree value by the user

// Draw the text at particular bounds.

args.Graphics.DrawString(headerValues[args.CellIndex], new PdfStandardFont(PdfFontFamily.Helvetica,
10), brush, new PointF(0, 0));

if (args.IsHeaderRow)
{
grid.Headers[0].Cells[args.CellIndex].Value = string.Empty;
}

args.Graphics.Restore();
}

private void PdfHeaderQueryCellInfo(object pdf)
{
ServerPdfHeaderQueryCellInfoEventArgs name = (ServerPdfHeaderQueryCellInfoEventArgs)pdf;
PdfGrid grid = new PdfGrid();
headerValues.Add(name.Column.HeaderText);
var longestString = headerValues.Where(s => s.Length == headerValues.Max(m => m.Length)).First();
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 6);
SizeF size = font.MeasureString(longestString);
name.Headers[0].Height = size.Width * 2;
}

```

Limitations

- The export feature for detail templates is not supported in server-side exporting.
- Multiple grids exporting feature is not supported with server side exporting.

Excel Export

Excel exporting in `##Platform_Name##` Grid control

The excel export allows exporting Grid data to Excel document. You need to use the [excelExport](#) method for exporting. To enable Excel export in the grid, set the [allowExcelExport](#) as true.

To use excel export, You need to inject the `ExcelExport` module in grid.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Page, ej.grids.ExcelExport, ej.grids.Toolbar);
var grid = new ej.grids.Grid({
  dataSource: data,

```

```

        allowPaging: true,
        allowExcelExport: true,
        toolbar: ['ExcelExport'],
        columns: [
            { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
            { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
            { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C' },
            { field: 'OrderDate', headerText: 'Order Date', width: 140, format:
'yMd', textAlign: 'Right' }
        ],
        height: 260
    });
    grid.toolbarClick = function(args){
        if (args['item'].id === 'Grid_excelexport') {
            grid.excelExport();
        }
    }
    grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

```

```

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Show spinner while exporting

You can show/ hide spinner component while exporting the grid using `showSpinner/ hideSpinner` methods. You can use `toolbarClick` event to show spinner before exporting and hide a spinner in the `pdfExportComplete` or `excelExportComplete` event after the exporting.

In the `toolbarClick` event, based on the parameter `args.item.id` as `Gridpdfexport` or `Gridexcelexport` we can call the `showSpinner` method from grid instance.

In the `pdfExportComplete` or `excelExportComplete` event, We can call the `hideSpinner` method.

In the below demo, we have rendered the default spinner component when exporting the grid.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Page, ej.grids.PdfExport,
ej.grids.ExcelExport, ej.grids.Toolbar);
var grid = new ej.grids.Grid({
    dataSource: data,

```

```

allowPaging: true,
allowPdfExport: true,
allowExcelExport: true,
toolbar: ['PdfExport', 'ExcelExport'],
columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
    { field: 'CustomerID', width: 140, headerText: 'Customer ID',
visible: false },
    { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120 },
    { field: 'ShipCity', headerText: 'ShipCity', textAlign: 'Right',
width: 140 }
],
height: 260
});
grid.appendTo('#Grid');
grid.toolbarClick = function(args) {
    if (args['item'].id === 'Grid_pdfexport') {
        grid.showSpinner();
        grid.pdfExport();
    }
    if (args['item'].id === 'Grid_excelext') {
        grid.showSpinner();
        grid.excelExport();
    }
}
grid.pdfExportComplete = () => {
    grid.hideSpinner();
}
grid.excelExportComplete = () => {
    grid.hideSpinner();
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Custom data source

The excel export provides an option to define datasource dynamically before exporting. To export data dynamically, define the `dataSource` in `exportProperties`.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Page, ej.grids.ExcelExport, ej.grids.Toolbar);
var grid = new ej.grids.Grid({
  dataSource: data,
  allowPaging: true,
  allowExcelExport: true,
  toolbar: ['ExcelExport'],
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
    { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
    { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C' },
    { field: 'OrderDate', headerText: 'Order Date', width: 140, format:
'yMd', textAlign: 'Right' }
  ],
  height: 315
});
grid.toolbarClick = function(args) {
  if (args['item'].id === 'Grid_excelexport') {
    var excelExportProperties = {
      dataSource: data
    };
    grid.excelExport(excelExportProperties);
  }
}
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Passing additional parameters to the server when exporting

You can pass the additional parameter in the `query` property by invoking `addParams` method. In the `toolbarClick` event, you can define params as key and value pair so it will receive at the server side when exporting.

In the below example, we have passed `recordcount` as `12` using `addParams` method.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Page, ej.grids.PdfExport,
ej.grids.ExcelExport, ej.grids.Toolbar);

```

```

var queryClone;
var grid = new ej.grids.Grid({
  dataSource: data,
  allowPaging: true,
  allowPdfExport: true,
  allowExcelExport: true,
  toolbar: ['PdfExport', 'ExcelExport'],
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
    { field: 'CustomerID', width: 140, headerText: 'Customer ID',
visible: false },
    { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120 },
    { field: 'ShipCity', headerText: 'ShipCity', textAlign: 'Right',
width: 140 }
  ],
  height: 260
});
grid.appendTo('#Grid');
grid.toolbarClick = function(args) {
  if (args['item'].id === 'Grid_pdfexport') {
    queryClone = grid.query;
    grid.query = new ej.data.Query().addParams("recordcount", "12");
    grid.pdfExport();
  }
  if (args['item'].id === 'Grid_excelexport') {
    queryClone = grid.query;
    grid.query = new ej.data.Query().addParams("recordcount", "12");
    grid.excelExport();
  }
}
grid.pdfExportComplete = () => {
  grid.query = queryClone;
}
grid.excelExportComplete = () => {
  grid.query = queryClone;
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js" type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Passing the selected records to the server using ajax request via custom toolbar button click

You can pass the selected records to the server with the help of an ajax request. In the `toolbarClick` event, you can get the selected records using the [getSelectedRecords](#) method and pass the selected records to the server using the **data** property of the ajax.

```
`ts
```

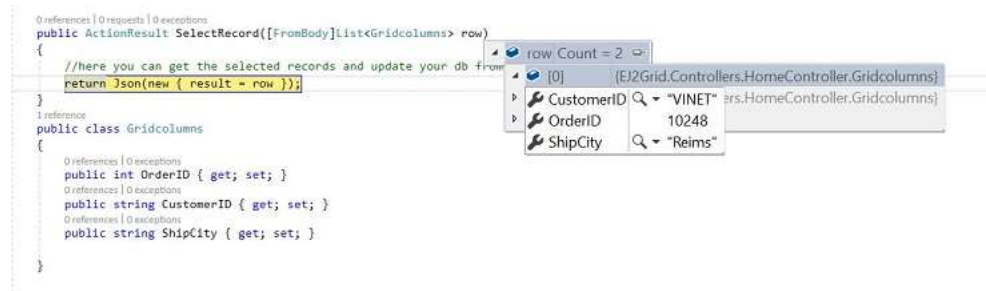
```
import { Grid, Page, Toolbar } from '@syncfusion/ej2-grids';
import { Ajax } from '@syncfusion/ej2-base';
Grid.Inject(Page, Toolbar);
let grid: Grid = new Grid({
  dataSource: data,
  allowSelection: true,
  toolbar: ['Selected'],
  columns: [
    { type: 'checkbox', width: 50, },
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right', width: 120},
    { field: 'CustomerID', width: 140, headerText: 'CustomerID'},
    { field: 'ShipCity', headerText: 'Ship Country', width: 140 }
  ],
  toolbarClick: function (args) {
    if (args.item.text === 'Selected') {
      var selectedRecord = this.getSelectedRecords();
      var ajax = new Ajax({
        url: "/Home/SelectRecord",
        type: "POST",
        contentType: "application/json",
        data: JSON.stringify(selectedRecord),
        successHandler: function (response) {
          console.log(JSON.parse(response));
        },
        failure: function (response) {
          alert(response);
        }
      });
      ajax.send();
    }
  }
});
```

```

}
},
});
grid.appendTo('#Grid');
`

```

The selected record details are bound to the **row** parameter. Please refer to the following screenshot.



[See Also](#)

- [Exporting Grid in Cordova application](#)

Export multiple grids in ##Platform_Name## Grid control

The Excel export provides an option to export multiple grid data in the same or different sheets of an Excel file. Each grid is identified by its unique ID. You can specify which grids to export by listing their IDs in the **exportGrids** property.

Same sheet

Excel exporting provides support for exporting multiple grids on the same sheet. To export the grids in the same sheet, define **multipleExport.type** as **AppendToSheet** in **exportProperties**. It also has an option to provide blank rows between the grids. These blank rows count can be defined by using **multipleExport.blankRows** property.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.ExcelExport, ej.grids.Page, ej.grids.Toolbar);
var firstGrid = new ej.grids.Grid({
    dataSource: data.slice(0, 5),
    allowPaging: true,
    allowExcelExport: true,
    exportGrids: ['FirstGrid', 'SecondGrid'],
    toolbar: ['ExcelExport'],
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
        { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
        { field: 'ShipName', headerText: 'Ship Name', width: 150 },
        { field: 'ShipCountry', headerText: 'Ship Country', width: 150 },
    ],
    height: 315
});

```

```

var secondGrid = new ej.grids.Grid({
  dataSource: employeeData.slice(0, 5),
  allowPaging: true,
  allowExcelExport: true,
  columns: [
    { field: 'EmployeeID', headerText: 'Employee ID', textAlign:
'Right', width: 120, type: 'number' },
    { field: 'FirstName', width: 140, headerText: 'First Name', type:
'string' },
    { field: 'LastName', width: 140, headerText: 'Last Name', type:
'string' },
    { field: 'City', headerText: 'City', width: 120 },
  ],
  height: 315
});
firstGrid.toolbarClick = function(args) {
  if (args['item'].id === 'FirstGrid_excelexport') {
    var appendExcelExportProperties = {
      multipleExport: { type: 'AppendToSheet', blankRows: 2 }
    };
    firstGrid.excelExport(appendExcelExportProperties, true);
  }
}
firstGrid.appendTo('#FirstGrid');
secondGrid.appendTo('#SecondGrid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <p><b>First Grid</b></p>
        <div id="FirstGrid"></div>
        <p><b>Second Grid</b></p>
        <div id="SecondGrid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

By default, `multipleExport.blankRows` value is 5.

[New sheet](#)

Excel export functionality enables the exporting of multiple grids onto separate sheets (each grid in new sheet of excel) within the Excel file. To achieve this, you can specify `multipleExport.type` as `NewSheet` in `exportProperties`.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.ExcelExport, ej.grids.Page, ej.grids.Toolbar);
var firstGrid = new ej.grids.Grid({
    dataSource: data.slice(0, 5),
    allowPaging: true,
    allowExcelExport: true,
    exportGrids: ['FirstGrid', 'SecondGrid'],
    toolbar: ['ExcelExport'],
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
        { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
        { field: 'ShipName', headerText: 'Ship Name', width: 150 },
        { field: 'ShipCountry', headerText: 'Ship Country', width: 150 },
    ],
    height: 315
});
var secondGrid = new ej.grids.Grid({

```

```

        dataSource: employeeData.slice(0, 5),
        allowPaging: true,
        allowExcelExport: true,
        columns: [
            { field: 'EmployeeID', headerText: 'Employee ID', textAlign:
'Right', width: 120, type: 'number' },
            { field: 'FirstName', width: 140, headerText: 'First Name', type:
'string' },
            { field: 'LastName', width: 140, headerText: 'Last Name', type:
'string' },
            { field: 'City', headerText: 'City', width: 120 },
        ],
        height: 315
    });
    firstGrid.toolbarClick = function(args) {
        if (args['item'].id === 'FirstGrid_excelexport') {
            var appendExcelExportProperties = {
                multipleExport: { type: 'NewSheet' }
            };
            firstGrid.excelExport(appendExcelExportProperties, true);
        }
    }
    firstGrid.appendTo('#FirstGrid');
    secondGrid.appendTo('#SecondGrid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <p><b>First Grid</b></p>
        <div id="FirstGrid"></div>
        <p><b>Second Grid</b></p>
        <div id="SecondGrid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Excel export options in ##Platform_Name## Grid control

The excel export provides an option to customize mapping of the grid to excel document.

Export current page

The excel export provides an option to export the current page into excel. To export current page, define `exportType` to `CurrentPage`.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Page, ej.grids.ExcelExport, ej.grids.Toolbar);
var grid = new ej.grids.Grid({
    dataSource: data,
    allowPaging: true,
    allowExcelExport: true,
    toolbar: ['ExcelExport'],
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
        { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C' },
        { field: 'OrderDate', headerText: 'Order Date', width: 140, format:
'yMd', textAlign: 'Right' }
    ],
    height: 315
});

```

```

grid.toolbarClick = function(args){
    if (args['item'].id === 'Grid_excelexport') {
        var excelExportProperties = {
            exportType: 'CurrentPage'
        };
        grid.excelExport(excelExportProperties);
    }
}
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <style>
        .e-row[aria-selected="true"] .e-customizedExpandcell {
            background-color: #e0e0e0;
        }
        .e-grid.e-gridhover tr[role='row']:hover {
            background-color: #eee;
        }
        .e-expand::before {
            content: '\e5b8';
        }
        .empImage {

```



```

        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Export the selected records only

You can export the selected records data by passing it to `exportProperties.dataSource` Property in the `toolbarClick` event.

In the below exporting demo, We can get the selected records using `getSelectedRecords` method and pass the selected data to `PdfExport` or `excelExport` property.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.PdfExport, ej.grids.ExcelExport,
ej.grids.Page, ej.grids.Toolbar, ej.grids.Filter);
var grid = new ej.grids.Grid({
    dataSource: data,
    allowPdfExport: true,
    allowExcelExport: true,
    allowPaging: true,
    allowFiltering: true,
    selectionSettings: { type: 'Multiple' },
    toolbar: ['PdfExport', 'ExcelExport'],
    pageSettings: { pageCount: 5, pageSize: 5 },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', width: 120,
        textAlign: 'Right' },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'OrderDate', headerText: 'Order Date', width: 130,
        format: 'yMd', textAlign: 'Right' },
        { field: 'Freight', width: 120, format: 'C2', textAlign: 'Right'
        },
        { field: 'ShipCountry', visible: false, headerText: 'Ship
        Country', width: 150 }
    ]
});

```

```

    ],
    });
grid.appendTo('#Grid');
grid.toolbarClick = (args) => {
    if (args.item.id === 'Grid_pdfexport') {
        let pdfdata = grid.getSelectedRecords();
        let exportProperties = {
            dataSource: pdfdata,
        };
        grid.pdfExport(exportProperties);
    }
    else if (args.item.id === 'Grid_excelexport') {
        let exceldata = grid.getSelectedRecords();
        let exportProperties = {
            dataSource: exceldata,
        };
        grid.excelExport(exportProperties);
    }
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <style>

```

```

        .e-row[aria-selected="true"] .e-customizedExpandcell {
            background-color: #e0e0e0;
        }
        .e-grid.e-gridhover tr[role='row']:hover {
            background-color: #eee;
        }
        .e-expand::before {
            content: '\e5b8';
        }
        .empImage {
            margin: 6px 16px;
            float: left;
            width: 50px;
            height: 50px;
        }
    </style>
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
    </head>
    <body>

        <div id="container">
            <div id="Grid"></div>
        </div>
    <script>
    var ele = document.getElementById('container');
    if(ele) {
        ele.style.visibility = "visible";
    }
    </script>
    <script src="index.js" type="text/javascript"></script>
    </body></html>

```

Export hidden columns

The excel export provides an option to export hidden columns of grid by defining `includeHiddenColumn` as `true`.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Page, ej.grids.ExcelExport, ej.grids.Toolbar);
var grid = new ej.grids.Grid({
    dataSource: data,
    allowPaging: true,
    allowExcelExport: true,
    toolbar: ['ExcelExport'],
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
        { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
        { field: 'ShipCountry', width: 140, headerText: 'Ship Country',
visible: false },
    ]
});

```

```

        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C' },
        { field: 'OrderDate', headerText: 'Order Date', width: 140, format:
'yMd', textAlign: 'Right' }
    ],
    height: 315
});
grid.toolbarClick = function(args) {
    if (args['item'].id === 'Grid_excelexport') {
        var excelExportProperties = {
            includeHiddenColumn: true
        };
        grid.excelExport(excelExportProperties);
    }
}
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <style>
        .e-row[aria-selected="true"] .e-customizedExpandcell {
            background-color: #e0e0e0;
        }
    </style>

```

```

        .e-grid.e-gridhover tr[role='row']:hover {
            background-color: #eee;
        }
        .e-expand::before {
            content: '\e5b8';
        }
        .empImage {
            margin: 6px 16px;
            float: left;
            width: 50px;
            height: 50px;
        }
    </style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Show or hide columns

You can show a hidden column or hide a visible column while printing the grid using [toolbarClick](#) and [excelExportComplete](#) events.

In the `toolbarClick` event, based on `args.item.id` as `Grid_excelexport`. We can show or hide columns by setting `column.visible` property to `true` or `false` respectively.

In the `excelExportComplete` event, We have reversed the state back to the previous state.

In the below example, we have `CustomerID` as a hidden column in the grid. While exporting, we have changed `CustomerID` to visible column and `ShipCity` as hidden column.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Page, ej.grids.ExcelExport, ej.grids.Toolbar);
var grid = new ej.grids.Grid({
    dataSource: data,
    allowPaging: true,
    allowPdfExport: true,
    toolbar: ['ExcelExport'],
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },

```

```

        { field: 'CustomerID', width: 140, headerText: 'Customer ID',
        visible: false },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
        width: 120 },
        { field: 'ShipCity', headerText: 'ShipCity', textAlign: 'Right',
        width: 140 }
    ],
    height: 260
});
grid.appendTo('#Grid');
grid.toolbarClick = function(args){
    if (args['item'].id === 'Grid_pdfexport') {
        grid.columns[1].visible = true;
        grid.columns[3].visible = false;
        grid.pdfExport();
    }
}
grid.excelExportComplete = () => {
    grid.columns[1].visible = false;
    grid.columns[3].visible = true;
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

```

```

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Export with filter options

The excel export provides an option to export with filter option in excel by defining `enableFilter` as `true`.

It requires the [allowFiltering](#) to be true.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Page, ej.grids.ExcelExport, ej.grids.Toolbar,
ej.grids.Filter);
var grid = new ej.grids.Grid({
    dataSource: data,
    allowPaging: true,
    allowExcelExport: true,
    allowFiltering: true,
    toolbar: ['ExcelExport'],
    columns: [

```

```

        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
        { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C' },
        { field: 'OrderDate', headerText: 'Order Date', width: 140, format:
'yMd', textAlign: 'Right' }
    ],
    height: 315
});
grid.toolbarClick = function(args){
    if (args['item'].id === 'Grid_excelexport') {
        var excelExportProperties = {
            enableFilter: true
        };
        grid.excelExport(excelExportProperties);
    }
}
grid.appendTo('#Grid');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
```



```

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Exporting grouped records

The excel export provides outline option for grouped records which hides the detailed data for better viewing. In grid, we have provided the outline option for the exported document when the data's are grouped.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Page, ej.grids.ExcelExport, ej.grids.Toolbar,
ej.grids.Group);
var grid = new ej.grids.Grid({
    dataSource: data,
    allowPaging: true,
    allowExcelExport: true,
    toolbar: ['ExcelExport'],
    allowGrouping: true,
    groupSettings: { columns: ['OrderID', 'CustomerID']},
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },

```

```

        { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
        'string' },
        { field: 'ShipCountry', width: 140, headerText: 'Ship Country',
        visible: false },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
        width: 120, format: 'C' },
        { field: 'OrderDate', headerText: 'Order Date', width: 140, format:
        'yMd', textAlign: 'Right' }
    ],
    height: 315
});
grid.toolbarClick = function(args) {
    if (args['item'].id === 'Grid_excelexport') {
        grid.excelExport();
    }
}
grid.appendTo('#Grid');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
    base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
    grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
    buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
    popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
    richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
    navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
    dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
    lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
    inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
    calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
    notifications/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
    splitbuttons/styles/material.css" rel="stylesheet">

    <style>
        .e-row[aria-selected="true"] .e-customizedExpandcell {
            background-color: #e0e0e0;
```

```

    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
}
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Define file name

You can assign the file name for the exported document by defining `fileName` property in [ExcelExportProperties](#).

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Page, ej.grids.ExcelExport, ej.grids.Toolbar);
let grid = new ej.grids.Grid({
    dataSource: data,
    allowPaging: true,
    allowExcelExport: true,
    toolbar: ['ExcelExport'],
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
        { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C' },
        { field: 'OrderDate', headerText: 'Order Date', width: 140, format:
'yMd', textAlign: 'Right' }
    ],

```

```

        height: 230
    });
    grid.toolbarClick = (args) => {
        if (args['item'].id === 'Grid_excelexport') {
            let excelExportProperties = {
                fileName: "new.xlsx"
            };
            grid.excelExport(excelExportProperties);
        }
    }
    grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <style>
        .e-row[aria-selected="true"] .e-customizedExpandcell {
            background-color: #e0e0e0;
        }
        .e-grid.e-gridhover tr[role='row']:hover {
            background-color: #eee;
        }
        .e-expand::before {
            content: '\e5b8';

```

```

    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Export the master detail grid

It is possible to export the master-detail grid on the same Excel sheet using the `ExcelExportProperties` class in the grid.

To export the master-detail grid on the same sheet in the following sample, you need to set the `multipleExport.type` to `AppendToSheet` in the `exportProperties`. A promise object is created by exporting the master grid first, and then the detail grid is exported after the master grid has been successfully exported.

INDEX.JS

```

var names = ['AROUT', 'BERGS', 'BLONP', 'CHOPS', 'ERNSH'];
var masterdata = customerData.filter(function(e) {
    return names.indexOf(e.CustomerID) !== -1;
});
var mastergrid = new ej.grids.Grid({
    dataSource: masterdata,
    allowExcelExport: true,
    toolbar: ['ExcelExport'],
    selectedRowIndex: 1,
    toolbarClick: toolbarClick,
    columns: [
        { field: 'ContactName', headerText: 'Customer Name', width: 150 },
        { field: 'CompanyName', headerText: 'Company Name', width: 150 },
        { field: 'Address', headerText: 'Address', width: 150 },
        { field: 'Country', headerText: 'Country', width: 130 },
    ],
    rowSelected: rowSelected,

```

```

});
mastergrid.appendTo('#MasterGrid');
function rowSelected(args) {
    let selectedRecord = args.data;
    grid.dataSource = data.filter((record) => record.CustomerName ===
selectedRecord.ContactName).slice(0, 5);
    document.getElementById('key').innerHTML = selectedRecord.ContactName;
}
function toolbarClick(args) {
    if (args.item.id === 'MasterGrid_excelexport') {
        const appendExcelExportProperties = {
            multipleExport: { type: 'AppendToSheet', blankRows: 2 },
        };
        const firstGridExport =
mastergrid.excelExport(appendExcelExportProperties, true);
        firstGridExport.then((fData) => {
            grid.excelExport(appendExcelExportProperties, false, fData);
        });
    }
}
var grid = new ej.grids.Grid({
    allowSelection: false,
    allowExcelExport: true,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', width: 100, textAlign:
'Right' },
        { field: 'Freight', headerText: 'Freight', width: 100, format: 'C2',
type: 'number' },
        { field: 'ShipName', headerText: 'Ship Name', width: 200 },
        { field: 'ShipCountry', headerText: 'Ship Country', width: 150 },
        { field: 'ShipAddress', headerText: 'Ship Address', width: 200 },
    ],
});
grid.appendTo('#DetailGrid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <p class="e-mastertext">Master Grid</p>
        <div id="MasterGrid">
        </div>
        <p><p class="e-mastertext">Detail Grid<p></p>
        <div id="DetailGrid">
        </div>
    </p><p></div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Export grid as blob

The Grid offers an option to export the data as a **Blob** instead of downloading it as a file in the browser. To export the grid as a Blob, set the **isBlob** parameter to **true** in the [excelExport](#) method. The grid returns the promise of a blob in the [excelExportComplete](#) event.

The following example demonstrates how to obtain the blob data of the exported grid by executing the promise in the [excelExportComplete](#) event.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Toolbar, ej.grids.ExcelExport, ej.grids.Page);
var grid = new ej.grids.Grid({
    dataSource: data,
    allowPaging: true,
    allowExcelExport: true,
    toolbar: ['ExcelExport', 'CsvExport'],

```

```

        toolbarClick: toolbarClick,
        excelExportComplete: excelExportComplete,
        columns: [
            { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
            { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
            { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C' },
            { field: 'OrderDate', headerText: 'Order Date', textAlign: 'Right',
width: 140, format: 'yMd' }
        ],
        height: 230
    });
    grid.appendTo('#Grid');
    function toolbarClick(args) {
        if (args.item.id === 'Grid_excelexport') {
            // pass fourth parameter as true to get the blob data of exported
grid
            grid.excelExport(null, null, null, true);
        }
        if (args.item.id === 'Grid_csvexport') {
            // pass fourth parameter as true to get the blob data of exported
grid
            grid.csvExport(null, null, null, true);
        }
    }
    function excelExportComplete(args) {
        // execute the promise to get the blob data
        args.promise.then((e) => {
            console.log(e.blobData);
        });
    };
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">

```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Excel cell style customization in ##Platform_Name## Grid control

Conditional cell formatting

Grid cells in the exported Excel can be customized or formatted using [excelQueryCellInfo](#) event. In this event, we can format the grid cells of exported PDF document based on the column cell value.

In the below sample, we have set the background color for **Freight** column in the exported excel by **args.cell** and **BackColor** property.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Page, ej.grids.ExcelExport, ej.grids.Toolbar);
var grid = new ej.grids.Grid({
    dataSource: data,
    allowPaging: true,
    allowPdfExport: true,
    toolbar: ['ExcelExport'],
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
        { field: 'CustomerID', width: 140, headerText: 'Customer ID', width:
120 },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120 },

```

```

        { field: 'ShipCity', headerText: 'ShipCity', textAlign: 'Right',
width: 140 }
    ],
    height: 260
});
grid.appendTo('#Grid');
grid.toolbarClick = function(args){
    if (args['item'].id === 'Grid_excelexport') {
        grid.excelExport();
    }
}
grid.excelQueryCellInfo = (args) => {
    if(args.column.field == 'Freight'){
        if(args.value < 30) {
            args.style = {backgroundColor: '#99ffcc'};
        }
        else if(args.value < 60) {
            args.style = {backgroundColor: '#ffffb3'};
        }
        else {
            args.style = {backgroundColor: '#ff704d'};
        }
    }
}
grid.queryCellInfo = (args) => {
    if(args.column.field == 'Freight'){
        if(args.data['Freight'] < 30) {
            args.cell.bgColor = '#99ffcc';
        }
        else if(args.data['Freight'] < 60) {
            args.cell.bgColor = '#ffffb3';
        }
        else {
            args.cell.bgColor = '#ff704d';
        }
    }
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js" type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Theme

The excel export provides an option to include theme for exported excel document.

To apply theme in exported Excel, define the **theme** in **exportProperties**.

INDEX.JS

```
ej.grids.Grid.Inject(ej.grids.Page, ej.grids.ExcelExport, ej.grids.Toolbar);
var grid = new ej.grids.Grid({
  dataSource: data,
  allowPaging: true,
  allowExcelExport: true,
  toolbar: ['ExcelExport'],
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
    { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
    { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C' },
    { field: 'OrderDate', headerText: 'Order Date', width: 140, format:
'yMd', textAlign: 'Right' }
  ],
  height: 315
});
grid.toolbarClick = function(args){
  if (args['item'].id === 'Grid_excelexport') {
    var excelExportProperties = {
      theme:
      {
        header: { fontName: 'Segoe UI', fontColor: '#666666' },
        record: { fontName: 'Segoe UI', fontColor: '#666666' },
        caption: { fontName: 'Segoe UI', fontColor: '#666666' }
      }
    };
    grid.excelExport(excelExportProperties);
  }
}
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js" type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

By default, material theme is applied to exported excel document.

Rotate a header text to a certain degree in the exported grid

The DataGrid has support to customize the column header styles such as changing text orientation, the font color, and so on in the exported Excel file. To achieve this requirement, use the [excelHeaderQueryCellInfo](#) event of the Grid.

The `excelHeaderQueryCellInfo` will be triggered when creating a column header for the excel document to be exported. Customize the column header in this event.

In the following demo, using the `rotation` property of the style argument in the `excelHeaderQueryCellInfo` event, you can rotate the header text of the column header in the excel exported document.

INDEX.JS

```
var grid = new ej.grids.Grid({
  dataSource: data,
  allowPaging: true,
  allowExcelExport: true,
  toolbar: ['ExcelExport'],
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
    { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string', customAttributes: { class: 'orientationcss' }, textAlign: 'Center'
},
    { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120 },
    { field: 'OrderDate', headerText: 'Order Date', textAlign: 'Right',
width: 140, format: 'yMd' }
  ],
  created: setHeaderHeight,
  height: 260
});

function setHeaderHeight(args) {
  var textWidth = document.querySelector(".orientationcss >
div").scrollWidth;
  var headerCell = document.querySelectorAll(".e-headercell");
  for (var i = 0; i < headerCell.length; i++) {
    headerCell.item(i).style.height = textWidth + 'px';
  }
}

grid.excelHeaderQueryCellInfo = function (args) {
  var textWidth = document.querySelector(".orientationcss >
div").scrollWidth;
  if (args.gridCell.column.field == 'Freight') {
    args.style = { backColor: '#99ffcc', vAlign: 'Bottom' };
  }
  else {
    args.style = { vAlign: 'Center', rotation: dropDownListObject.value
};
  }
  args.cell.cellHeight = textWidth;
};

grid.excelQueryCellInfo = function (args) {
  if (args.column.field == 'Freight') {
```

```

        if (args.value < 30) {
            args.style = { backColor: '#99ffcc' };
        }
        else if (args.value < 60) {
            args.style = { backColor: '#ffffb3' };
        }
        else {
            args.style = { backColor: '#ff704d' };
        }
    }
};
grid.toolbarClick = function (args) {
    if (args['item'].id === 'Grid_excelexport') {
        grid.excelExport();
    }
};
grid.appendTo('#Grid');
var degree = [90, 180, 45, 135, 225, -90];
var dropDownListObject = new ej.dropdowns.DropDownList({
    dataSource: degree,
    placeholder: "Select a degree"
});
dropDownListObject.appendTo('#ddlelement');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-calendars/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="toolbar-template">
            <div id="dropdown" style="margin-top:5px">
                <input type="text" tabindex="1" id="ddlelement">
            </div>
        </div>
        <div id="Grid"></div>
    </div>
<style>
    .orientationcss .e-headercelldiv {
        transform: rotate(90deg);
    }
</style>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Adding header and footer in ##Platform_Name## Grid control

The excel export provides an option to include header and footer content for exported excel document.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Page, ej.grids.ExcelExport, ej.grids.Toolbar);
var grid = new ej.grids.Grid({
    dataSource: data,
    allowPaging: true,
    allowExcelExport: true,
    toolbar: ['ExcelExport'],
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
        { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C' },
        { field: 'OrderDate', headerText: 'Order Date', width: 140, format:
'yMd', textAlign: 'Right' }
    ],
    height: 315

```



```

});
grid.toolbarClick = function(args){
    if (args['item'].id === 'Grid_excelexport') {
        var excelExportProperties = {
            header: {
                headerRows: 7,
                rows: [
                    { cells: [{ colSpan: 4, value: "Northwind Traders",
style: { fontColor: '#C67878', fontSize: 20, hAlign: 'Center', bold: true, }
}] },
                    { cells: [{ colSpan: 4, value: "2501 Aerial Center
Parkway", style: { fontColor: '#C67878', fontSize: 15, hAlign: 'Center',
bold: true, } }] },
                    { cells: [{ colSpan: 4, value: "Suite 200 Morrisville,
NC 27560 USA", style: { fontColor: '#C67878', fontSize: 15, hAlign:
'Center', bold: true, } }] },
                    { cells: [{ colSpan: 4, value: "Tel +1 888.936.8638 Fax
+1 919.573.0306", style: { fontColor: '#C67878', fontSize: 15, hAlign:
'Center', bold: true, } }] },
                    { cells: [{ colSpan: 4, hyperlink: { target:
'https://www.northwind.com/', displayText: 'www.northwind.com' }, style: {
hAlign: 'Center' } }] },
                    { cells: [{ colSpan: 4, hyperlink: { target:
'mailto:support@northwind.com' }, style: { hAlign: 'Center' } }] },
                ]
            },
            footer: {
                footerRows: 4,
                rows: [
                    { cells: [{ colSpan: 4, value: "Thank you for your
business!", style: { hAlign: 'Center', bold: true } }] },
                    { cells: [{ colSpan: 4, value: "!Visit Again!", style: {
hAlign: 'Center', bold: true } }] }
                ]
            },
        };
        grid.excelExport(excelExportProperties);
    }
}
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Exporting hierarchy grid in ##Platform_Name## Grid control

The grid has an option to export the hierarchy grid to excel document. By default, it will use **Expanded** as hierarchyExportMode. you can change the exporting option by using the **ExcelExportProperties.hierarchyExportMode** property. The available options are,

| Mode | Behavior |

|-----|-----|

| Expanded | Exports the visible child grids in expanded state and remaining child grid in collapsed state when args.isChild property is set to true in [beforeExcelExport](#) event. |

| All | Exports the all the child grids in expanded state. |

| None | Exports all child grids in collapsed state when args.isChild property is set to true in [beforeExcelExport](#) event. |

INDEX.JS

```
ej.grids.Grid.Inject(ej.grids.DetailRow, ej.grids.Toolbar,
ej.grids.ExcelExport);
var grid= new ej.grids.Grid({
  dataSource: employeeData,
  toolbar: ["ExcelExport"],
  allowExcelExport: true,
  columns: [
    { field: 'EmployeeID', headerText: 'Employee ID', textAlign:
'Right', width: 120 },
    { field: 'FirstName', headerText: 'First Name', width: 150 },
    { field: 'City', headerText: 'City', width: 150 },
    { field: 'Country', headerText: 'Country', width: 150 }
  ],
  childGrid: {
    dataSource: data,
    queryString: 'EmployeeID',
    columns: [
      { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
      { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
      { field: 'ShipCity', headerText: 'Ship City', width: 150 },
      { field: 'ShipName', headerText: 'Ship Name', width: 150 }
    ],
  },
  beforeExcelExport: beforeExcelExport,
  toolbarClick: toolbarClick
});
grid.appendTo('#Grid');
function beforeExcelExport(args) {
  args.isChild = true;
}
function toolbarClick(args) {
  if (args['item'].id === 'Grid_excelexport') {
    var exportProperties = {
      hierarchyExportMode: "Expanded"
    };
    grid.excelExport(exportProperties);
  }
}
```

```
}
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Grid"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
```

```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Limitations

- Microsoft Excel permits up to seven nested levels in outlines. So that in the grid we can able to provide only up to seven nested level

and if it exceeds more than seven levels then the document will be exported without outline option. Please refer the [Microsoft Limitation](#)

Exporting grid with templates in ##Platform_Name## Grid control

The grid offers the option to export the column, detail, and caption templates to an Excel document. The template contains images, hyperlinks, and customized text.

Exporting with column template

The Excel export functionality allows you to export Grid columns that include images, hyperlinks, and custom text to an Excel document.

In the following sample, the hyperlinks and images are exported to Excel using [hyperlink](#) and [image](#) properties in the [excelQueryCellInfo](#) event.

Excel Export supports base64 string to export the images.

INDEX.JS

```
ej.grids.Grid.Inject(ej.grids.Toolbar, ej.grids.ExcelExport);
var grid = new ej.grids.Grid({
  dataSource: employeeData,
  allowExcelExport: true,
  toolbar: ['ExcelExport'],
  columns: [
    {
      headerText: 'Employee Image', textAlign: 'Center',
      template: '#imageTemplate', width: 150
    },
    { field: 'EmployeeID', headerText: 'Employee ID', textAlign:
'Right', width: 125 },
    { field: 'FirstName', headerText: 'Name', width: 120 },
    { field: 'EmailID', headerText: 'Email ID', template:
'#mailTemplate', width: 170 }
  ],
  toolbarClick: toolbarClick,
  excelQueryCellInfo: excelQueryCellInfo,
  height: 273
});
grid.appendTo('#Grid');
function toolbarClick(args) {
  if (args.item.id === 'Grid_excelexport') {
    grid.excelExport();
  }
}
function excelQueryCellInfo(args) {
  if (args.column.headerText === 'Employee Image') {
    args.image = {
```

```

        base64: args.data.EmployeeImage,
        height: 70,
        width: 70,
    };
}
if (args.column.headerText === 'Email ID') {
    args.hyperLink = {
        target: 'mailto:' + args.data.EmailID,
        displayText: args.data.EmailID,
    };
}
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">





  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```

```
<div id="container">
  <script id="imageTemplate" type="text/x-template">
    <div class="image">
      
    </div>
  </script>
  <script id="mailTemplate" type="text/x-template">
    <div class="link">
      <a href="mailto:${EmailID}">${EmailID}</a></div>
    </div>
  </script>
  <div id="Grid"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Excel Export		
Employee Image	Name	Email ID
	Nancy	nancy@domain.com
	Andrew	andrew@domain.com
	Janet	janet@domain.com
	Margaret	margaret@domain.com

Exporting with detail template

By default, the grid will export the parent grid with expanded detail rows alone. Change the exporting option by using the `ExcelExportProperties.hierarchyExportMode` property. The available options are:

Mode	Behavior
----- -----	
Expanded	Exports the parent grid with expanded detail rows.
All	Exports the parent grid with all the detail rows.
None	Exports the parent grid alone.

The detail rows in the exported Excel can be customized or formatted using the [exportDetailTemplate](#) event. In this event, the detail rows of the Excel document are formatted in accordance with their parent row details.

In the following sample, the detail row content is formatted by specifying the [columnHeader](#) and [rows](#) properties using its [parentRow](#) details. This allows for the creation of detail rows in the Excel document. Additionally, custom styles can be applied to specific cells using the [style](#) property.

When using [rowSpan](#), it is essential to provide the cell's [index](#) for proper functionality.

INDEX.JS

```
ej.grids.Grid.Inject(ej.grids.DetailRow, ej.grids.Toolbar,
ej.grids.ExcelExport);
var grid= new ej.grids.Grid({
  dataSource: employeeData,
  detailTemplate: '#detailtemplate',
  toolbar: ['ExcelExport'],
  allowExcelExport: true,
  toolbarClick: toolbarClick,
  exportDetailTemplate: exportDetailTemplate,
  columns: [
    { field: 'Category', headerText: 'Category', width: 120 },
    { field: 'ProductID', headerText: 'Product ID', width: 140 },
    { field: 'Status', headerText: 'Status', width: 200 }
  ],
  height: 273
});
grid.appendTo('#Grid');
function toolbarClick(args) {
  if (args['item'].id === 'Grid_excelexport') {
    var exportProperties = {
      hierarchyExportMode: "Expanded"
    };
    grid.excelExport(exportProperties);
  }
}
function exportDetailTemplate(args) {
  args.value = {
    columnHeader: [
      {
        cells: [{
          index: 0, colSpan: 2, value: 'Product Details',
```



```

        style: { backgroundColor: '#ADD8E6', excelHAlign: 'Center',
bold: true }
        ]]
    },
    ],
    rows: [
        {
            cells: [
                {
                    index: 0, rowspan: 4, image: {
                        base64: args.parentRow.data['ProductImg'],
                        height: 80, width: 100
                    }
                },
                {
                    index: 1, value: "Offers: " +
args.parentRow.data['Offers'],
                    style: { bold: true, fontColor: '#0a76ff' }
                },
            ]
        },
        {
            cells: [
                {
                    index: 1, value: 'Available: ' +
args.parentRow.data['Available']
                }
            ]
        },
        {
            cells: [
                {
                    index: 1, value: 'Contact: ',
                    hyperlink: {
                        target: 'mailto:' +
args.parentRow.data['Contact'],
                        displayText: args.parentRow.data['Contact']
                    }
                }
            ]
        },
        {
            cells: [
                {
                    index: 1, value: 'Ratings: ' +
args.parentRow.data['Ratings'],
                    style: { bold: true, fontColor: '#0a76ff' }
                }
            ]
        },
        {
            cells: [
                {
                    index: 0, value: args.parentRow.data['productDesc'],
                    style: { excelHAlign: 'Center' }
                },
                { index: 1, value: args.parentRow.data['ReturnPolicy'] }
            ]
        },
    ],

```

```

        {
            cells: [
                {
                    index: 0, value: args.parentRow.data['Cost'],
                    style: { excelHAlign: 'Center', bold: true }
                },
                { index: 1, value: args.parentRow.data['Cancellation'] }
            ]
        },
        {
            cells: [
                {
                    index: 0, value: args.parentRow.data['Status'],
                    style: {
                        bold: true, fontColor:
args.parentRow.data['Status'] === 'Available' ? '#00FF00' : '#FF0000',
                        excelHAlign: 'Center'
                    }
                },
                {
                    index: 1, value: args.parentRow.data['Delivery'],
                    style: { bold: true, fontColor: '#0a76ff' }
                }
            ]
        }
    ],
};
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <script id="detailtemplate">
    <table class="detailtable" width="100%">
      <colgroup>
        <col width="40%" />
        <col width="60%" />
      </colgroup>
      <thead>
        <tr>
          <th colspan="2" style="font-weight: 500;text-align:
center;background-color: #ADD8E6;">
            Product Details
          </th>
        </tr>
      </thead>
      <tbody>
        <tr>
          <td rowspan="4" style="text-align: center;">
            
          </td>
          <td>
            <span style="font-weight: 500;color:
#0a76ff;">Offers: {Offers} </span>
          </td>
        </tr>
        <tr>
          <td>
            <span>Available: {Available} </span>
          </td>
        </tr>
        <tr>
          <td>
            <span class="link">
              Contact: <a
href="mailto:{Contact}">{Contact}</a>
            </span>
          </td>
        </tr>
        <tr>
          <td>

```

```

        <span style="font-weight: 500;color: #0a76ff;">
Ratings: ${Ratings}</span>
        </td>
    </tr>
    <tr>
        <td style="text-align: center;">
            <span> ${productDesc}</span>
        </td>
        <td>
            <span>${ReturnPolicy}</span>
        </td>
    </tr>
    <tr>
        <td style="text-align: center;">
            <span style="font-weight: 500;" > ${Cost}</span>
        </td>
        <td>
            <span>${Cancellation}</span>
        </td>
    </tr>
    <tr>
        <td style="text-align: center;">
            <span class="${Status}" style="font-weight: 500;" >
${Status}</span>
        </td>
        <td>
            <span style="font-weight: 500;color:
#0a76ff;">${Delivery}</span>
        </td>
    </tr>
</tbody>
</table>
</script>
<div id="container">
    <div id="Grid"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Excel Export			
	Category	Product ID	Status
▶	Suits/Slim	EJ-SU-01	Available
▶	Suits/Classic	EJ-SU-02	Available
▶	Suits/Formal	EJ-SU-03	Available
▶	Phants/Slim	EJ-PH-01	Available
▶	Phants/Classic	EJ-PH-02	Available
▶	Shirts/Slim	EJ-SH-01	Available
▶	Shirts/Formal	EJ-SH-02	Available

Exporting with caption template

The Excel export feature enables exporting of Grid with a caption template to an Excel document.

In the following sample, the customized caption text is exported to Excel using [captionText](#) property in the [exportGroupCaption](#) event.

INDEX.JS

```
ej.grids.Grid.Inject(ej.grids.Group, ej.grids.Toolbar,
ej.grids.ExcelExport);
var grid = new ej.grids.Grid({
  dataSource: employeeData,
  allowGrouping: true,
  groupSettings: { captionTemplate: '#captiontemplate', columns:
[ 'EmployeeID' ] },
  allowExcelExport: true,
  toolbar: [ 'ExcelExport' ],
  columns: [
    { field: 'EmployeeID', headerText: 'Employee ID', width: 120 },
    { field: 'FirstName', headerText: 'Name', width: 120 },
    { field: 'City', headerText: 'City' },
    { field: 'Title', headerText: 'Title', width: 170 }
  ],
  toolbarClick: toolbarClick,
  exportGroupCaption: exportGroupCaption,
  height: 273
});
```

```

});
grid.appendTo('#Grid');
function toolbarClick(args) {
    if (args.item.id === 'Grid_excelexport') {
        grid.excelExport();
    }
}

function exportGroupCaption(args) {
    args.captionText = args.data.field + ' - ' + args.data.key;
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

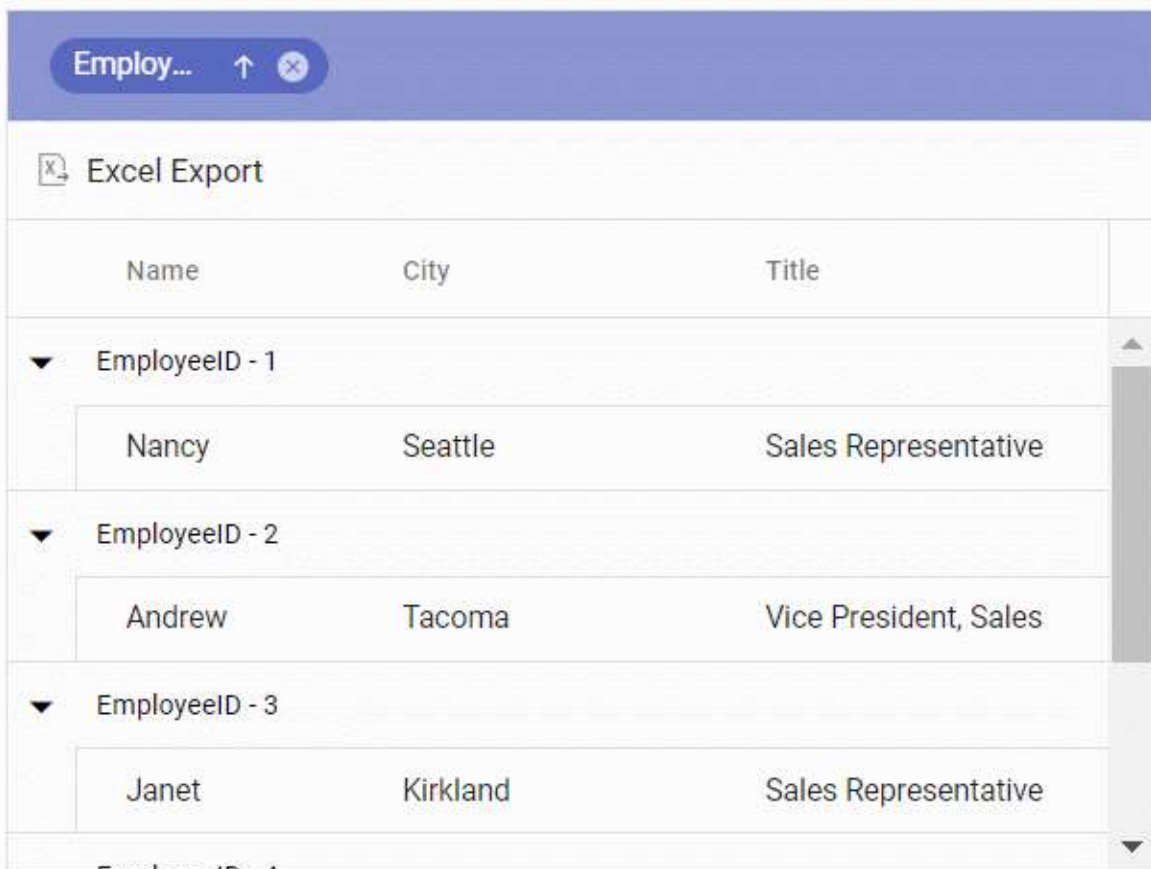
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```

```

<div id="container">
  <script id="captiontemplate" type="text/x-template">
    ${field} - ${key}
  </script>
  <div id="Grid"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```



The screenshot shows a web interface with a blue header bar containing a button labeled "Employ..." with an upward arrow and a close icon. Below the header is a white section with a tab labeled "Excel Export". The main content area displays a table with three columns: "Name", "City", and "Title". The table contains three rows of data, each preceded by a dropdown arrow and a label: "EmployeeID - 1", "EmployeeID - 2", and "EmployeeID - 3".

	Name	City	Title
▼ EmployeeID - 1	Nancy	Seattle	Sales Representative
▼ EmployeeID - 2	Andrew	Tacoma	Vice President, Sales
▼ EmployeeID - 3	Janet	Kirkland	Sales Representative

Exporting grid in server in ##Platform_Name## Grid control

The Grid have an option to export the data to Excel in server side using Grid server export library.

Server dependencies

The Server side export functionality is shipped in the Syncfusion.EJ2.GridExport package, which is available in Essential Studio and nuget.org. The following list of dependencies is required for Grid server side Excel exporting action.

- Syncfusion.EJ2

- Syncfusion.EJ2.GridExport
- Syncfusion.Compression.Base
- Syncfusion.XlsIO.Base

Server configuration

The following code snippets shows server configuration using ASP.NET MVC Controller Action.

To Export the Grid in server side, You need to call the [serverExcelExport](#) method for passing the Grid properties to server exporting action.

```
`ts
public ActionResult ExcelExport(string gridModel)
{
    GridExcelExport exp = new GridExcelExport();
    Grid gridProperty = ConvertGridObject(gridModel);
    return exp.ExcelExport<OrdersDetails>(gridProperty, OrderRepository.GetAllRecords());
}

private Grid ConvertGridObject(string gridProperty)
{
    Grid GridModel = (Grid)Newtonsoft.Json.JsonConvert.DeserializeObject(gridProperty, typeof(Grid));
    GridColumnModel cols =
    (GridColumnModel)Newtonsoft.Json.JsonConvert.DeserializeObject(gridProperty,
    typeof(GridColumnModel));
    GridModel.Columns = cols.columns;
    return GridModel;
}

public class GridColumnModel
{
    public List<GridColumn> columns { get; set; }
}

public ActionResult DataSource(DataManager dm)
{
    var DataSource = OrderRepository.GetAllRecords();
    DataResult result = new DataResult();
    result.result = DataSource.Skip(dm.Skip).Take(dm.Take).ToList();
    result.count = result.result.Count;
    return Json(result, JsonRequestBehavior.AllowGet);
}
```



```

,
`ts
import { Grid, Toolbar } from '@syncfusion/ej2-grids';
import { DataManager, UrlAdaptor } from '@syncfusion/ej2-data';
Grid.Inject(Toolbar);
let data: DataManager = new DataManager({
url: "Home/DataSource",
adaptor: new UrlAdaptor
});
let grid: Grid = new Grid({
dataSource: data,
toolbar: ['ExcelExport'],
columns: [
{ field: 'OrderID', headerText: 'Order ID', textAlign: 'Right', width: 100 },
{ field: 'CustomerID', headerText: 'Customer ID', width: 120 },
{ field: 'Freight', headerText: 'Freight', textAlign: 'Right', width: 120, format: 'C2' },
{ field: 'ShipCountry', headerText: 'Ship Country', width: 150 }
],
height: 265
});
grid.appendTo('#Grid');
grid.toolbarClick = (args: Object) => {
if (args['item'].id === 'Grid_excelexport') {
grid.serverExcelExport("Home/ExcelExport");
}
}
,

```

Note: Refer to the GitHub sample for quick implementation and testing from [here](#).

CSV export in server side

You can export the Grid to CSV format by using the [serverCsvExport](#) method which will pass the Grid properties to server.

In the below demo, we have invoked the above method inside the [toolbarClick](#) event. In server side, we have deserialized the Grid properties and passed to the `CsvExport` method which will export the properties to CSV format.

```

`ts
public ActionResult CsvGridExport(string gridModel)
{
    GridExcelExport exp = new GridExcelExport();
    Grid gridProperty = ConvertGridObject(gridModel);
    return exp.CsvExport<OrdersDetails>(gridProperty, OrderRepository.GetAllRecords());
}

private Grid ConvertGridObject(string gridProperty)
{
    Grid GridModel = (Grid)Newtonsoft.Json.JsonConvert.DeserializeObject(gridProperty, typeof(Grid));
    GridColumnModel cols =
    (GridColumnModel)Newtonsoft.Json.JsonConvert.DeserializeObject(gridProperty,
    typeof(GridColumnModel));
    GridModel.Columns = cols.columns;
    return GridModel;
}

public ActionResult DataSource(DataManager dm)
{
    var DataSource = OrderRepository.GetAllRecords();
    DataResult result = new DataResult();
    result.result = DataSource.Skip(dm.Skip).Take(dm.Take).ToList();
    result.count = result.result.Count;
    return Json(result, JsonRequestBehavior.AllowGet);
}
`

`ts
import { Grid, Toolbar } from '@syncfusion/ej2-grids';
import { DataManager, UrlAdaptor } from '@syncfusion/ej2-data';
Grid.Inject(Toolbar);
let data: DataManager = new DataManager({
    url: "Home/DataSource",
    adaptor: new UrlAdaptor
});
let grid: Grid = new Grid({

```

```

dataSource: data,
toolbar: ['CsvExport'],
columns: [
{ field: 'OrderID', headerText: 'Order ID', textAlign: 'Right', width: 100 },
{ field: 'CustomerID', headerText: 'Customer ID', width: 120 },
{ field: 'Freight', headerText: 'Freight', textAlign: 'Right', width: 120, format: 'C2' },
{ field: 'ShipCountry', headerText: 'Ship Country', width: 150 }
],
height: 265
});
grid.appendTo('#Grid');
grid.toolbarClick = (args: Object) => {
if (args['item'].id === 'Grid_csvexport') {
grid.serverCsvExport("Home/CsvGridExport");
}
}
`

```

Export grid as memory stream

The Grid offers an option to export the data as a memory stream instead of downloading it as a file in the browser. To obtain the memory stream of the exported grid, set the `AsMemoryStream` parameter to **true** in the [ExcelExport](#) and [CsvExport](#) methods.

The following code demonstrates how to get the memory stream of exported grid.

```

`ts
public object ExcelExport(string gridModel)
{
GridExcelExport exp = new GridExcelExport();
Grid gridProperty = ConvertGridObject(gridModel);
// pass third parameter as true to get the Memory Stream of exported grid data
return (MemoryStream)exp.ExcelExport<OrdersDetails>(gridProperty, OrderRepository.GetAllRecords(),
true);
}
public object CsvExport(string gridModel)
{
GridExcelExport exp = new GridExcelExport();

```

```

Grid gridProperty = ConvertGridObject(gridModel);

return (MemoryStream)exp.CsvExport<OrdersDetails>(gridProperty, OrderRepository.GetAllRecords(),
true);
}
`ts

```

Merge grid's memory stream

The [Essential XlsIO](#) library is used to merge multiple memory streams into a single stream. To learn more about the merge option, please refer to this [documentation](#).

You can merge a memory stream, a file stream, and a local file with the Grid's memory stream in the following ways:

Merging with an existing memory stream

If you already have a memory stream, you can directly use it to merge with the Grid's memory stream.

In the following code, **ExcelEngine** and **AddCopy** method of Worksheets are used to merge the grid's memory stream with an existing memory stream.

```

`ts
using Syncfusion.XlsIO;

public MemoryStream ms1; // defines existing memory stream

public object ExcelExport(string gridModel)
{
    GridExcelExport exp = new GridExcelExport();
    Grid gridProperty = ConvertGridObject(gridModel);
    // get the memory stream of exported grid data
    MemoryStream ms2 = (MemoryStream)exp.ExcelExport<OrdersDetails>(gridProperty,
    OrderRepository.GetAllRecords(), true);

    //New instance of ExcelEngine is created equivalent to launching Microsoft Excel with no workbooks
    open
    ExcelEngine excelEngine = new ExcelEngine();
    //Instantiate the Excel application object
    IApplication application = excelEngine.Excel;
    //Assigns default application version
    application.DefaultVersion = ExcelVersion.Xlsx;
    //open an workbook of existing memory stream and grid's memory stream through Open method of
    IWorkbooks
    IWorkbook sourceWorkbook = application.Workbooks.Open(ms1);
    IWorkbook destinationWorkbook = application.Workbooks.Open(ms2);
    //Copy all the worksheet from the Source workbook to the destination workbook

```

```

for (int i = 0; i < sourceWorkbook.Worksheets.Count; i++)
{
    destinationWorkbook.Worksheets.AddCopy(sourceWorkbook.Worksheets[i]);
}
destinationWorkbook.ActiveSheetIndex = 1;
//Saving the workbook as stream
MemoryStream ms3 = new MemoryStream();
destinationWorkbook.SaveAs(ms3);
ms3.Position = 0;
//Dispose the instance of ExcelEngine
excelEngine.Dispose();
//Dispose the streams.
ms1.Dispose();
ms2.Dispose();
return ms3;
}
`

```

Merging with an existing file stream

If you already have a file stream, you can directly use it to merge with the Grid's memory stream. In the following code, the existing file stream is merged with the Grid's memory stream.

```

`ts
using Syncfusion.XlsIO;

public FileStream fs1; // defines existing file stream

public object ExcelExport(string gridModel)
{
    GridExcelExport exp = new GridExcelExport();
    Grid gridProperty = ConvertGridObject(gridModel);
    MemoryStream ms1 = (MemoryStream)exp.ExcelExport<OrdersDetails>(gridProperty,
    OrderRepository.GetAllRecords(), true);
    ExcelEngine excelEngine = new ExcelEngine();
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Xlsx;
    //fs1 and ms1 represents the existing stream and grid's stream.
    IWorkbook sourceWorkbook = application.Workbooks.Open(fs1);

```

```

IWorkbook destinationWorkbook = application.Workbooks.Open(ms1);
for (int i = 0; i < sourceWorkbook.Worksheets.Count; i++)
{
    destinationWorkbook.Worksheets.AddCopy(sourceWorkbook.Worksheets[i]);
}
destinationWorkbook.ActiveSheetIndex = 1;
//Saving the workbook as stream
MemoryStream ms3 = new MemoryStream();
destinationWorkbook.SaveAs(ms3);
ms3.Position = 0;
return ms3;
}
`

```

Merging with a local file

To merge a local file with the Grid's memory stream, you need to convert it into a file stream before merging. In the following code, the existing local file is merged with the Grid's memory stream.

```

`ts
using Syncfusion.XlsIO;

// get the file stream of local file
public FileStream fs1 = new FileStream("D:/ExcelDoc.xlsx", FileMode.Open, FileAccess.Read); //
ExcelDoc.xlsx is a local file which is located in local disk D.

public object ExcelExport(string gridModel)
{
    GridExcelExport exp = new GridExcelExport();
    Grid gridProperty = ConvertGridObject(gridModel);
    MemoryStream ms1 = (MemoryStream)exp.ExcelExport<OrdersDetails>(gridProperty,
    OrderRepository.GetAllRecords(), true);
    ExcelEngine excelEngine = new ExcelEngine();
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Xlsx;
    //fs1 and ms1 represents the local file's stream and grid's stream.
    IWorkbook sourceWorkbook = application.Workbooks.Open(fs1);
    IWorkbook destinationWorkbook = application.Workbooks.Open(ms1);
    for (int i = 0; i < sourceWorkbook.Worksheets.Count; i++)

```

```

{
destinationWorkbook.Worksheets.AddCopy(sourceWorkbook.Worksheets[i]);
}
destinationWorkbook.ActiveSheetIndex = 1;
MemoryStream ms3 = new MemoryStream();
destinationWorkbook.SaveAs(ms3);
ms3.Position = 0;
return ms3;
}
`

```

[Downloading the merged memory stream](#)

You can download the merged memory stream by converting it into a `FileStreamResult`. In the following code, the merged memory stream is downloaded to the browser.

```

`ts
using Syncfusion.XlsIO;

public ActionResult ExcelExport(string gridModel)
{
ExcelEngine excelEngine = new ExcelEngine();
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Xlsx;
//open an workbook of streams through Open method of IWorkbooks
IWorkbook sourceWorkbook = application.Workbooks.Open(ms1);
IWorkbook destinationWorkbook = application.Workbooks.Open(ms2);
for (int i = 0; i < sourceWorkbook.Worksheets.Count; i++)
{
destinationWorkbook.Worksheets.AddCopy(sourceWorkbook.Worksheets[i]);
}
destinationWorkbook.ActiveSheetIndex = 1;
MemoryStream ms3 = new MemoryStream();
destinationWorkbook.SaveAs(ms3);
ms3.Position = 0;
// Save the MemoryStream into FileStreamResult
FileStreamResult fileStreamResult = new FileStreamResult(ms3, "Application/vnd.openxmlformats-officedocument.spreadsheetml.sheet");

```

```

fileStreamResult.FileDownloadName = "Export.xlsx";
//Dispose the instance of ExcelEngine
excelEngine.Dispose();
//Dispose the streams.
ms1.Dispose();
ms2.Dispose();
// return the file
return fileStreamResult;
}

```

Rotate a header text to a certain degree in the exported grid on the server side

The DataGrid has support to customize the column header styles such as changing text orientation, the font color, and so on in the exported Excel file. To achieve this requirement, use the `ServerExcelHeaderQueryCellInfo` event of the Grid.

The `ServerExcelHeaderQueryCellInfo` will be triggered when creating a column header for the excel document to be exported in the server side. Customize the column header in this event.

In the following demo, using the `HeaderCellRotate` method of the `GridExcelExport` class in the `ServerExcelHeaderQueryCellInfo` event, you can rotate the header text of the column header in the excel exported document.

```

`ts
public ActionResult ExcelExport(string gridModel)
{
    GridExcelExport exp = new GridExcelExport();
    Grid gridProperty = ConvertGridObject(gridModel);
    gridProperty.ServerExcelHeaderQueryCellInfo = ExcelHeaderQueryCellInfo;
    IEnumerable data = Utils.DataTableToJson(dt);
    var result = exp.ExcelExport<dynamic>(gridProperty, data);
    return result;
}

private void ExcelHeaderQueryCellInfo(object excel)
{
    ServerExcelHeaderQueryCellInfoEventArgs name = (ServerExcelHeaderQueryCellInfoEventArgs)excel;
    headerValues.Add(name.Column.HeaderText);
    var longestString = headerValues.Where(s => s.Length == headerValues.Max(m => m.Length)).First();
    GridExcelExport exp = new GridExcelExport();

```



```

var size = exp.ExcelTextSize(name.Style.Font.FontName, (float)name.Style.Font.Size, longestString);
name.Cell.RowHeight = size.Width;
exp.HeaderCellRotate(name, 45); // Give the rotate degree value by the user.
name.Style.Borders.LineStyle = Syncfusion.XlsIO.ExcelLineStyle.None;
}
`

```

Limitations

- The export feature for detail templates is not supported in server-side exporting.
- Multiple grids exporting feature is not supported with server side exporting.

Global local in ##Platform_Name## Grid control

Localization

The [Localization](#) library allows you to localize default text content of the Grid. The grid component has static text on some features (like group drop area text, pager information text, etc.) that can be changed to other cultures (Arabic, Deutsch, French, etc.) by defining the [locale](#) value and translation object.

The following list of properties and its values are used in the grid.

Locale keywords | Text

EmptyRecord | No records to display

True | true

False | false

InvalidFilterMessage | Invalid Filter Data

GroupDropArea | Drag a column header here to group its column

UnGroup | Click here to ungroup

GroupDisable | Grouping is disabled for this column

FilterbarTitle | \s filter bar cell

EmptyDataSourceError | DataSource must not be empty at initial load since columns are generated from dataSource in AutoGenerate Column Grid

Add | Add

Edit | Edit

Cancel | Cancel

Update | Update

Delete | Delete

Print | Print

Pdfexport | PDF Export

Exceleexport | Excel Export

Wordexport | Word Export

Csvexport | CSV Export

Search | Search

Columnchooser | Columns

Save | Save

Item | item

Items | items

EditOperationAlert | No records selected for edit operation

DeleteOperationAlert | No records selected for delete operation

SaveButton | Save

OKButton | OK

CancelButton | Cancel

EditFormTitle | Details of

AddFormTitle | Add New Record

BatchSaveConfirm | Are you sure you want to save changes?

BatchSaveLostChanges | Unsaved changes will be lost. Are you sure you want to continue?

ConfirmDelete | Are you sure you want to Delete Record?

CancelEdit | Are you sure you want to Cancel the changes?

ChooseColumns | Choose Column

SearchColumns | search columns

Matches | No Matches Found

FilterButton | Filter

ClearButton | Clear

StartsWith | Starts With

EndsWith | Ends With

Contains | Contains

Equal | Equal

NotEqual | Not Equal

LessThan | Less Than

LessThanOrEqual | Less Than Or Equal

GreaterThan | Greater Than

GreaterThanOrEqual | Greater Than Or Equal

ChooseDate | Choose a Date

EnterValue | Enter the value

Copy | Copy

Group | Group by this column

Ungroup | Ungroup by this column

autoFitAll | AutoFit all columns

autoFit | AutoFit this column

Export | Export

FirstPage | First Page

LastPage | Last Page

PreviousPage | Previous Page

NextPage | Next Page

SortAscending | Sort Ascending

SortDescending | Sort Descending

EditRecord | Edit Record

DeleteRecord | Delete Record

FilterMenu | Filter

SelectAll | Select All

Blanks | Blanks

FilterTrue | True

FilterFalse | False

NoResult | No Matches Found

ClearFilter | Clear Filter

NumberFilter | Number Filters

TextFilter | Text Filters

DateFilter | Date Filters

MatchCase | Match Case

Between | Between

CustomFilter | Custom Filter

CustomFilterPlaceholder | Enter the value

CustomFilterDatePlaceholder | Choose a date

AND | AND

OR | OR

ShowRowsWhere | Show rows where:

currentPageInfo | {0} of {1} pages

totalItemsInfo | ({0} items)

totalItemInfo | ({0} item)

firstPageTooltip | Go to first page

lastPageTooltip | Go to last page

nextPageTooltip | Go to next page

previousPageTooltip | Go to previous page

nextPagerTooltip | Go to next pager items

previousPagerTooltip | Go to previous pager items

pagerDropDown | Items per page

pagerAllDropDown | Items

All | All

Loading translations

To load translation object in an application, use [load](#) function of the [L10n](#) class.

The following example demonstrates the Grid in **Deutsch** culture.

INDEX.JS

```
ej.base.L10n.load({
  'de-DE': {
    'grid': {
      'EmptyRecord': 'Keine Aufzeichnungen angezeigt',
      'GroupDropArea': 'Ziehen Sie einen Spaltenkopf hier, um die
Gruppe ihre Spalte',
      'UnGroup': 'Klicken Sie hier, um die Gruppierung aufheben',
      'EmptyDataSourceError': 'DataSource darf bei der Erstaustellung
nicht leer sein, da Spalten aus der dataSource im AutoGenerate
Spaltenraster',
      'Item': 'Artikel',
      'Items': 'Artikel'
    },
    'pager': {
      'currentPageInfo': '{0} von {1} Seiten',
      'totalItemsInfo': '({0} Beiträge)',
      'firstPageTooltip': 'Zur ersten Seite',
      'lastPageTooltip': 'Zur letzten Seite',
      'nextPageTooltip': 'Zur nächsten Seite',
      'previousPageTooltip': 'Zurück zur letzten Seit',
      'nextPagerTooltip': 'Gehen Sie zu den nächsten Pager-Elementen',
      'previousPagerTooltip': 'Gehen Sie zu vorherigen Pager-
Elementen'
    }
  }
});
ej.grids.Grid.Inject(ej.grids.Page, ej.grids.Group);
var grid = new ej.grids.Grid({
  dataSource: data,
  locale: 'de-DE',
```

```

allowGrouping: true,
allowPaging: true,
pageSettings: { pageSize: 6 },
columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
    { field: 'ShipCity', headerText: 'Ship City', width: 150 },
    { field: 'ShipName', headerText: 'Ship Name', width: 150 }
],
height: 220
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
<title>EJ2 Grid</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
.e-row[aria-selected="true"] .e-customizedExpandcell {
    background-color: #e0e0e0;
}
.e-grid.e-gridhover tr[role='row']:hover {
    background-color: #eee;
}

```

```

        .e-expand::before {
            content: '\e5b8';
        }
        .empImage {
            margin: 6px 16px;
            float: left;
            width: 50px;
            height: 50px;
        }
    </style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Internationalization

The [Internationalization](#) library is used to globalize number, date, and time values in grid component using format strings in the [columns.format](#).

INDEX.JS

```

ej.base.setCulture('de');
ej.base.setCurrencyCode('EUR');
ej.base.L10n.load({
    'de-DE': {
        'grid': {
            'EmptyRecord': 'Keine Aufzeichnungen angezeigt',
            'GroupDropArea': 'Ziehen Sie einen Spaltenkopf hier, um die
Gruppe ihre Spalte',
            'UnGroup': 'Klicken Sie hier, um die Gruppierung aufheben',
            'EmptyDataSourceError': 'DataSource darf bei der Erstausslastung
nicht leer sein, da Spalten aus der dataSource im AutoGenerate
Spaltenraster',
            'Item': 'Artikel',
            'Items': 'Artikel'
        },
        'pager': {
            'currentPageInfo': '{0} von {1} Seiten',
            'totalItemsInfo': '({0} Beiträge)',
            'firstPageTooltip': 'Zur ersten Seite',
            'lastPageTooltip': 'Zur letzten Seite',

```

```

        'nextPageTooltip': 'Zur nächsten Seite',
        'previousPageTooltip': 'Zurück zur letzten Seit',
        'nextPagerTooltip': 'Gehen Sie zu den nächsten Pager-Elementen',
        'previousPagerTooltip': 'Gehen Sie zu vorherigen Pager-
Elementen'
    }
}
});
ej.grids.Grid.Inject(ej.grids.Page, ej.grids.Group);
var grid = new ej.grids.Grid({
    dataSource: data,
    locale: 'de-DE',
    allowGrouping: true,
    allowPaging: true,
    pageSettings: { pageSize: 6 },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        {
            field: 'Freight', headerText: 'Freight', width: 150, format: {
                format: 'C2', useGrouping: false,
                minimumSignificantDigits: 1, maximumSignificantDigits: 3,
currency: 'EUR'
            }, textAlign: 'Right'
        },
        { field: 'ShipName', headerText: 'Ship Name', width: 150 }
    ],
    height: 220
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

* In the above sample, Freight column is formatted by NumberFormatOptions.

* By default, [locale](#) value is en-US. If you want to change the en-US culture to a different culture, you have to change the [locale](#) accordingly.

Right to left (RTL)

RTL provides an option to switch the text direction and layout of the Grid component from right to left. It improves the user experiences and accessibility for users who use right-to-left languages (Arabic, Farsi, Urdu, etc.). To enable RTL Grid, set the [enableRtl](#) to true.

INDEX.JS

```
ej.base.L10n.load({
  'ar-AE': {
    'grid': {
      'EmptyRecord': 'لا سجلات لعرضها',
      'EmptyDataSourceError': 'يجب أن يكون مصدر البيانات فارغة في',
      'التحميل الأولي منذ يتم إنشاء الأعمدة من مصدر البيانات في أوتوجينيراتد عمود الشبكة'
    },
    'pager': {
      'currentPageInfo': '{0} صفحة 1 {من}',
      'totalItemsInfo': '({0} العناصر)',
      'firstPageTooltip': 'انتقل إلى الصفحة الأولى',
      'lastPageTooltip': 'انتقل إلى الصفحة الأخيرة',
      'nextPageTooltip': 'انتقل إلى الصفحة التالية',
      'previousPageTooltip': 'انتقل إلى الصفحة السابقة',
      'nextPagerTooltip': 'انتقل إلى عناصر بيكر التالية',
      'previousPagerTooltip': 'للذهاب إلى عناصر بيكر السابقة'
    }
  }
});
ej.grids.Grid.Inject(ej.grids.Page);
var grid = new ej.grids.Grid({
  dataSource: data,
  enableRtl: true,
  locale: 'ar-AE',
  allowPaging: true,
  pageSettings: { pageSize: 7 },
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
    { field: 'ShipCity', headerText: 'Ship City', width: 150 },
    { field: 'ShipName', headerText: 'Ship Name', width: 150 }
  ],
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}

```

```

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Internationalization](#)
- [Localization](#)

Accessibility in ##Platform_Name## Grid control

The Grid component followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

WAI-ARIA attributes

The Grid component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Grid component:

Attributes	Purpose
---	---
<code>role=grid</code>	To represent the element containing the grid component.
<code>role=row</code>	To represent the element containing the cells of the row in the grid.
<code>role=rowgroup</code>	To represent the group of rows in the grid.
<code>role=columnheader</code>	To represent the cell in a row contains header information for a column in the grid.
<code>role=gridcell</code>	To represent a cell in the grid component.
<code>role=button</code>	To represent the element that acts as a button in the grid.
<code>role=search</code>	To represent the element that acts as a search region in the grid.
<code>role=presentation</code>	To represent the element to be not available for accessibility concerns.
<code>role=navigation</code>	To represent the element containing pager elements to navigate from one page to another.
<code>aria-colindex</code>	Defines the column index of the column with respect to the total number of columns within the grid.
<code>aria-rowindex</code>	Defines row index of the row with respect to the total number of rows within the grid.
<code>aria-rowspan</code>	Defines the number of rows spanned by a cell within the grid.
<code>aria-colspan</code>	Defines the number of columns spanned by a cell within the grid.
<code>aria-rowcount</code>	Defines the total number of rows in the grid.
<code>aria-colcount</code>	Defines the total number of columns in the grid.
<code>aria-selected</code>	Indicates the current "selected" state of the rows and cells in the grid.

| **aria-expanded** | Indicate if the expand icon in the hierarchy grid or grouped grid or detail grid is expanded or collapsed |

| **aria-sort** | Indicates whether the data in the grid are sorted in ascending or descending order. |

| **aria-busy** | Indicates an element is being modified and that assistive technologies may want to wait until the changes are complete before informing the user about the update. |

| **aria-owns** | Identifies an element in order to define a visual, functional, or contextual relationship between a parent and its child elements. |

| **aria-hidden** | Hides the element from accessibility concerns. |

| **aria-labelledby** | Provides an accessible name for the checkbox labels in excel filter, checkbox filter and column chooser dialog. |

| **aria-describedby** | Provides an description about the features enabled in the header when the grid header cell is focused. |

Keyboard interaction

The Grid component followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Grid component.

Pager

Windows | MAC | To do this

Home | Fn + Left Arrow | Moves the focus to the first cell of the focused row.

End | Fn + Right Arrow | Moves the focus to the last cell of the focused row.

Ctrl + Home | Command + Fn + Left Arrow | Moves the focus to the first Cell of the first row in the grid.

Ctrl + End | Command + Fn + Right Arrow | Moves the focus to the last Cell of the last row in the grid.

Up Arrow | Up Arrow | Moves the cell focus upward from the focused cell.

Down Arrow | Down Arrow | Moves the cell focus downward from the focused cell.

Right Arrow | Right Arrow | Moves the cell focus right side from the focused cell.

Left Arrow | Left Arrow | Moves the cell focus left side from the focused cell.

Alt + J | Alt + J | Moves the focus to the entire grid.

Alt + W | Alt + W | Move the focus to the grid content element.

Selection

Windows | MAC | To do this

Ctrl + Up Arrow | Command + Up Arrow | Collapses all the visible groups.

Ctrl + Down Arrow | Command + Down Arrow | Expands all the visible groups.

Ctrl + Space | Ctrl + Space | Performs grouping when focused on a header element.

Enter | Enter | If the current cell is an expand/collapse cell then expands/collapses the current group/detailrow/childgrid.

Print

Windows | MAC | To do this

Ctrl + C | Command + C | Copies selected rows or cells data into the clipboard.

Ctrl + Shift + H | Ctrl + Shift + H | Copies selected rows or cells data with header into clipboard

Editing

Windows | MAC | To do this

Alt + Down arrow | Alt + Down arrow | Opens the filter menu(excel, menu and checkbox filter) when its header element is in focused state.

Column Menu

Windows | MAC | To do this

Ctrl + left arrow or right arrow | Command + left arrow or right arrow | Reorders the focused header column to the left or right side.

Sorting

Windows | MAC | To do this

Enter | Enter | Performs sorting(ascending/descending) on a column when its header element is in focused state.

Ctrl + Enter | Command + Enter | Performs multi-sorting on a column when its header element is in focused state.

Shift + Enter | Shift + Enter | Clears sorting for the focused header column.

* The Command and Control keys on Mac devices can be interchanged. When this switch occurs, use the Command key in place of the Control key and the Control key in place of the Command key for the above listed key interactions with Mac devices.

* For example, after switching the keys to group the columns when the header element is focused use Command + Space and for expanding the visible groups use Ctrl + Down Arrow.

Ensuring accessibility

The Grid component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Grid component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Grid component with accessibility tools.

See also

- [Accessibility in Syncfusion ##Platform_Name## components](#)

